DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943-5002

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

ADAPTIVE CONTROL WITH FINITE TIME
PERSISTENCY OF EXCITATION

by

Irfan Onuk

June 1986

Thesis Advisor Roberto Cristi

Approved for public release; distribution is unlimited

T232229

UNCLASSIFIED
CURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
. SECURITY CLASSIFICATION AUTHORITY . DECLASSIFICATION / DOWNGRADING SCHEDULE	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)

. NAME OF PERFORMING ORGANIZATION aval Postgraduate School	6b OFFICE SYMBOL (If applicable) Code 62	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
ADDRESS (City, State, and ZIP Code) onterey, California 93943-5000		7b. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
NAME OF FUNDING / SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

ADDRESS (City, State, and ZIP Code)

10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

TITLE (Include Security Classification)
DAPTIVE CONTROL WITH FINITE TIME PERSISTENCY OF EXCITATION

PERSONAL AUTHOR(S)
huk, Irfan

. TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
aster's Thesis	FROM ______ TO ______	1986, June	129

SUPPLEMENTARY NOTATION

COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Adaptive Control Persistency of Excitation

ABSTRACT (Continue on reverse if necessary and identify by block number)

An indirect adaptive control algorithm has been studied to control hysical systems with parameter uncertainties. Although the particular lgorithm investigated is applicable to a wide class of discrete time inear systems, parameter convergence and therefore global stability of he whole system is guaranteed only if the external excitation contains a ufficiently large number of frequency components.

This research study has also investigated the possibility of stopping he identification procedure when the parameter error becomes sufficiently mall, so the controller automatically turns itself from adaptive to time nvariant, while still guaranteeing global stability of the closed-loop ystem. In this way the adaptive controller might be activated only when ts gains do not provide satisfactory performances.

DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
NAME OF RESPONSIBLE INDIVIDUAL oberto Cristi	22b TELEPHONE (Include Area Code) (408) 646-2223	22c OFFICE SYMBOL Code 62Cx

FORM 1473, 84 MAR
83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

#19 - ABSTRACT - (CONTINUED)

Computer programs of the indirect adaptive control have been written for simulation purposes using both recursive least squares and projection algorithms.

Approved for public release; distribution is unlimited.

Adaptive Control with Finite Time Persistency of Excitation

by

Irfan Onuk
Lieutenant Junior Grade, Turkish Navy
B.S., Turkish Naval Academy, 1979

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1986

Thesis
05[illegible]312
c.1

ABSTRACT

An indirect adaptive control algorithm has been studied to control physical systems with parameter uncertainties. Although the particular algorithm investigated is applicable to a wide class of discrete time linear systems, parameter convergence and therefore global stability of the whole system is guaranteed only if the external excitation contains a sufficiently large number of frequency components.

This research study has also investigated the possibility of stopping the identification procedure when the parameter error becomes sufficiently small, so the controller automatically turns itself from adaptive to time invariant, while still guaranteeing global stability of the closed-loop system. In this way the adaptive controller might be activated only when its gains do not provide satisfactory performances.

Computer programs of the indirect adaptive control have been written for simulation purposes using both recursive least squares and projection algorithms.

TABLE OF CONTENTS

I. INTRODUCTION -- 6

A. GAIN SCHEDULING ------------------------------ 7

B. MODEL-REFERENCE ADAPTIVE SYSTEMS ----------- 8

C. SELF-TUNING REGULATORS AND POLE PLACEMENT -- 10

II. ZEROES OF SAMPLED DATA SYSTEMS ----------------- 14

A. TIME DOMAIN ANALYSIS OF THE ZEROES OF SAMPLED DATA TRANSFER FUNCTIONS ------------ 14

B. FREQUENCY DOMAIN ANALYSIS OF THE ZEROES OF SAMPLED DATA SYSTEMS ----------------------- 22

III. INDIRECT ADAPTIVE CONTROL FOR DISCRETE TIME SYSTEMS -- 28

A. POLE PLACEMENT DESIGN BASED ON INPUT-OUTPUT MODELS ------------------------------ 30

B. DIOPHANTINE EQUATION ----------------------- 36

C. PARAMETER ESTIMATION ----------------------- 39

D. PERSISTENCY OF EXCITATION ------------------ 44

E. FINITE TIME PERSISTENCY OF EXCITATION ------ 46

IV. SIMULATION STUDIES ----------------------------- 50

APPENDIX A: DESCRIPTION OF THE COMPUTER PROGRAMS ---- 81

APPENDIX B: COMPUTER PROGRAM CONDIS ----------------- 83

APPENDIX C: COMPUTER PROGRAM RCIOR ------------------ 91

APPENDIX D: COMPUTER PROGRAM RCIOP ------------------ 109

LIST OF REFERENCES ----------------------------------- 125

INITIAL DISTRIBUTION LIST ---------------------------- 127

I. INTRODUCTION

One of the most challenging, interesting and active fields of Automatic Control is Adaptive Control. To implement high-performance control systems when the plant dynamics are poorly known or when large and unpredictable variations occur, the control engineers prefer to use an important class of control systems called Adaptive Control systems. Adaptive Control comes from a desire and need for improving performance of complex engineering systems with large uncertainties. It is especially important in systems with many unknown parameters that are changing with time. Also, it can be defined as a special type of nonlinear feedback control, as a nonlinear, nonautonomous dynamic system.

An adaptive controller can change its behavior in response to changes in the dynamics of the plant and the disturbances.

The term adaptive control has been used at least from the beginning of the 1950's. With recent advances in microprocessor technology, it has become feasible to implement adaptive algorithms efficiently in real time at a reasonable cost.

There are three schemes for parameter adaptive control: gain scheduling, model reference control and self-tuning regulators. The starting point is an ordinary feedback control loop with a process and regulator with adjustable parameters. But the main problem is to find a convenient way of changing the regulator parameters in response to changes in process and disturbance dynamics. The following schemes differ only in the way the parameters of the regulator are adjusted.

A. GAIN SCHEDULING

Gain scheduling depends on finding auxiliary process variables correlated with the changes in plant dynamics. In this way it is possible to reduce the effects of parameter variations by changing the parameters of the regulator as functions of the auxiliary variables.

The block diagram of this scheme is given in Figure 1.1.

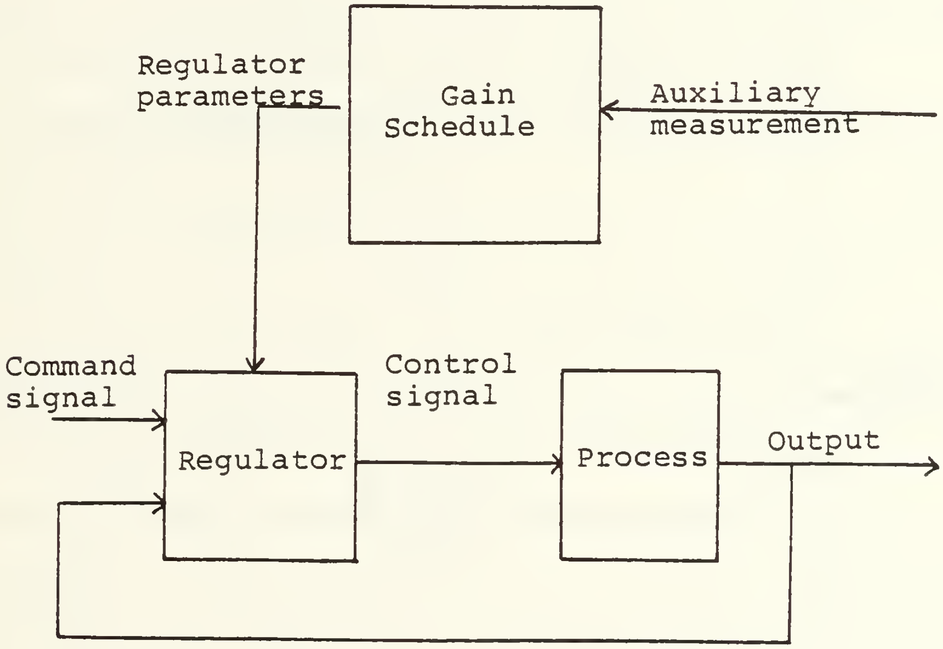

Figure 1.1. Block Diagram of Gain-Scheduling System

B. MODEL-REFERENCE ADAPTIVE SYSTEMS

In this type of adaptive system, a reference model specifies the desired performance, and tells how the process output should respond to the command signal. A block diagram of model reference system is given in Figure 1.2. As seen, from this figure, the reference model is part of the control system.

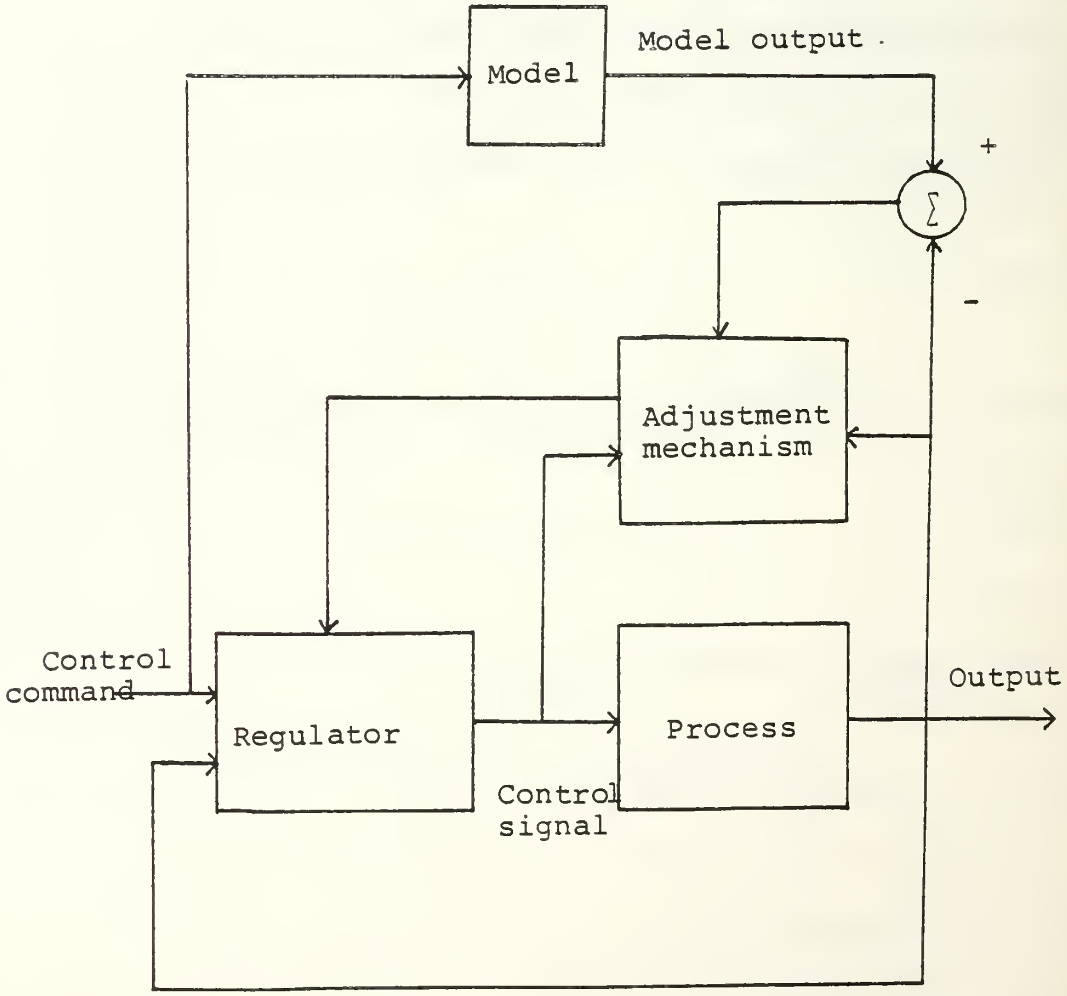

Figure 1.2. Block Diagram of the Model Reference Adaptive System

This adaptive system has two loops: the lower loop contains Regulator and Process. The upper loop adjusts the parameters of the regulator, in such a way that the error between the model output and the process output tends to zero. In this type of control system, the main problem is to determine the adjustment mechanism so that a stable system results, which brings the tracking error to zero.

Model reference adaptive systems can be further subdivided into two categories: direct and indirect. In indirect control the plant parameters are estimated and the control parameters are adjusted based on these estimates so that the overall plant transfer function matches that of the reference model. In direct control no effort is made to identify the plant parameters but the control parameters are directly adjusted to minimize the error between plant and model outputs.

It turns out that the model reference approach is applicable only to plants with stable zeroes. In fact the only way the closed loop transfer function (Plant & Regulator) can match the one of the model in its poles and zeroes, is by removing the plant zeroes by cancellation with closed loop poles. This operation leads to the presence of uncontrollable or unobservable modes in the closed loop systems, which can be accepted only if they are stable (i.e., their effect decays to zero with time), and this constitutes the major limitation for model reference adaptive systems.

C. SELF-TUNING REGULATORS AND POLE PLACEMENT

A third approach to the adaptive problem is the self-tuning regulator. A block diagram of this type control system is given in Figure 1.3.

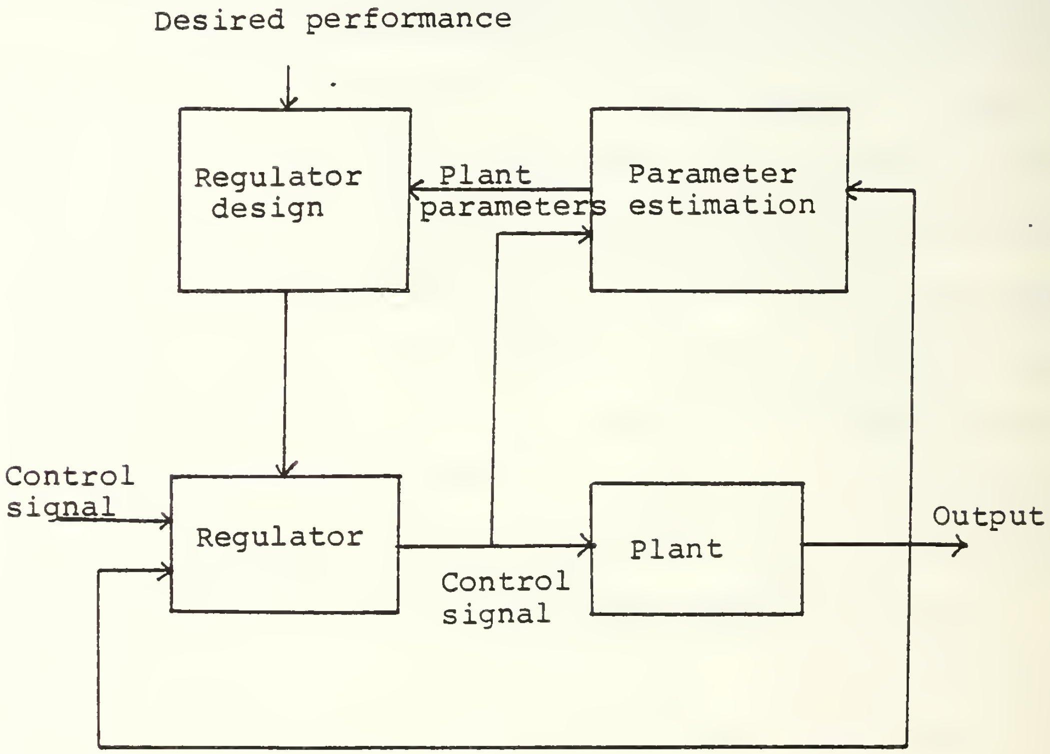

Figure 1.3. Block Diagram of a Self-Tuning Regulator

The reference model of the previous approach is replaced by some more general desired performances; such as error minimization or desired closed loop poles.

There are two loops in the system configuration. The lower loop contains the plant and linear feedback regulator.

The upper loop consists of a recursive parameter estimator and a design calculation adjusts the parameters of the regulator.

Also it is possible to classify this adaptive control scheme as direct and indirect. This classification depends on the complexity of the design calculation block that is seen in Figure 1.4.

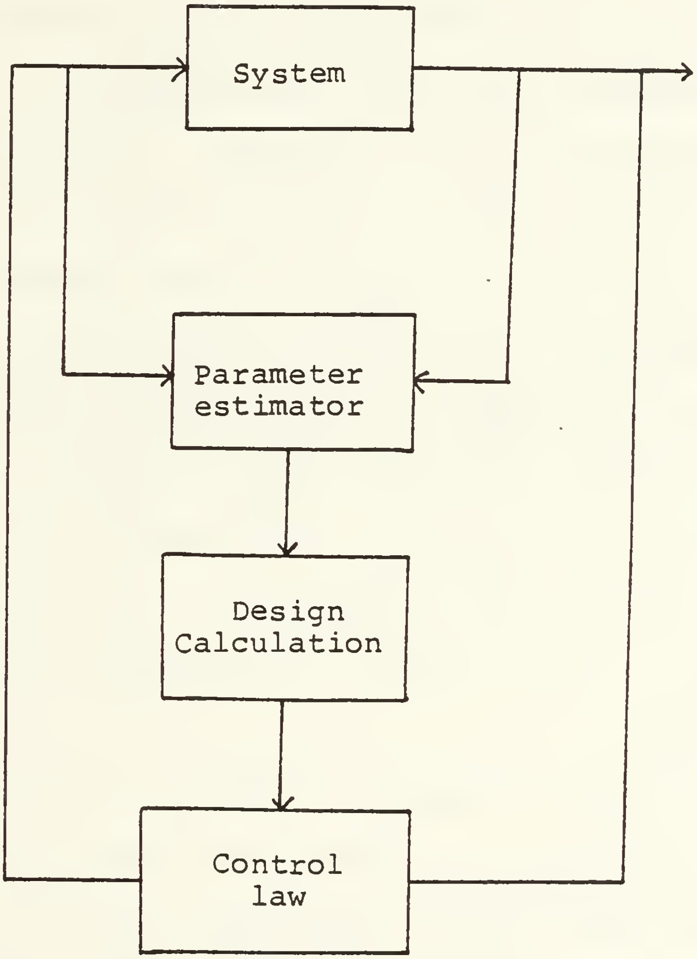

Figure 1.4. General Block Diagram of an Adaptive Control System

The first method is to parameterize the system directly in terms of compensator parameters.

This research project will concentrate on the last method, that is, pole placement and indirect adaptive control. The evaluation of the control law is indirectly determined on the basis of parameter estimates. This method is also called explicit, since the design is based on an explicit estimation of the process model.. The parameters of the controller are updated indirectly via estimation of the process parameters. Several algorithms to estimate the parameters of a linear model are available in the literature. In this thesis two well-known estimation algorithms will be investigated: Recursive Least Squares and Projection. They represent a tradeoff between complexity of computation and performances, in the sense that best performances (with relatively high complexity are obtained by using Recursive Least Squares. Also, Block-processing is investigated to determine the period of the adaptation of the control parameters. Finite time persistency of excitation is also studied and simulated for indirect adaptive control.

The main reason of choice of indirect adaptive control for this research comes from the fact that it is applicable to nonminimum-phase systems, and there are no limitations on zeroes of the plant. Therefore, it is more general than model reference adaptive control.

Because of the effects of the zeroes of sampled data systems on adaptive control theory, we start investigating the behavior of zeroes of sampled data systems. This thesis is organized as follows: In Chapter II zeroes of sampled data systems are analyzed by stating results available in the literature, and simulation studies. The indirect adaptive control problem is presented in Chapter III and computer simulation studies are given in Chapter IV.

The computer programs are presented in the Appendices B, C and D. Also the description of the programs is given in Appendix A.

II. ZEROES OF SAMPLED DATA SYSTEMS

Many control strategies are based on the assumption that the zeroes of the plant are on a stable region of the complex plane so that they can be cancelled by precompensation or closed loop poles. One example is the model reference adaptive control mentioned in the Introduction. However it turns out that in sampled data systems with Zero Order Hold (the most popular mean of Digital to Analog conversion) some zeroes of the resulting discrete time plant are often in the unstable region. In this chapter we analyze the position of the zeroes of sampled data systems, in relation with the continuous time plant dynamics and sampling frequency. The structure, the number of zeroes outside the unit disc and the low frequency characteristic of the pulse transfer function are examined from the transfer function for an important set of continuous time processes.

Finally, it is investigated that zeroes of the sampled data systems are sensitive to high frequency poles present in continuous time transfer function.

A. TIME DOMAIN ANALYSIS OF THE ZEROES OF SAMPLED DATA TRANSFER FUNCTIONS

Poles and zeroes are important parameters of linear time-invariant systems. The zeroes describe the way the internal variables are coupled to the inputs and the outputs. As described in [Ref. 6], the unstable zeroes limit the performance of control systems, since many design techniques are based on the cancellation of the process zeroes. This can be done provided the system's zeroes are stable.

The transformation of poles in the continuous time domain to the corresponding discrete time is given as:

$$P_i \rightarrow e^{P_i T} \qquad (2.1)$$

where T is the sampling period. This transformation maps the left-half part of the s-plane onto the unit disc, so that stability is preserved. But there is no simple transformation for zeroes from continuous to discrete time domain. The type of hold circuit affects the position of the zeroes. Most digital control systems use a zero-order hold, and for this type of hold circuit, the effects are considered.

In the following discussion, the main results are limit theorems, which give the zero locations for small and large sampling periods.

If the continuous time transfer function G(S) is rational it follows from Equation (2.2) that H(z) is also a rational function

$$H(z) = (1-z^{-1}) \sum_i ((e^{P_i T})/(z-e^{P_i T})) \mathrm{ReS}_{P_i} G(s)/s \qquad (2.2)$$

where:

$$\mathrm{ReS}_{P_i} = \lim_{s \to P_i} G(s)/s(s-P_i)$$

and P_i are the poles of G(s)/s.

The function H(z) has generically n-1 zeroes. For particular values of the sampling period some zeroes may, however, go to infinity, or they may be cancelled by poles, i.e., hidden modes. Hidden modes should not be considered as zeroes of H(z). In fact, there are in general no simple closed form expressions for the zeroes of H. The limiting cases for small or large sampling periods can be characterized. That is explained in the following theorem. The major steps in the proof are given by [Ref. 9].

Theorem 1:

Let G(s) be a rational function

$$G(s) = K \frac{(s-z_1)(s-z_2) \ \ldots \ (s-z_m)}{(s-p_1)(s-p_2) \ \ldots \ (s-p_n)} \qquad (2.3)$$

and H(z) the corresponding pulse transfer function. Assume that $m < n$. Then as the sampling period $T \to 0$, m zeroes of H(z) go to 1 as $\exp(z_i T)$ and the remaining n-m-1 zeroes of H(z) go to the zeroes of $B_{n-m}(z)$ where $B_n(z)$ is the polynomial defined as

$$B_n(z) = b_1 n_z^{n-1} + b_2 n_z^{n-2} + \ldots + b_n^n \qquad (2.4)$$

and

$$b_k^n = \sum_{\ell=1}^{k} (-1)^{k-1} \ell^n \binom{n+\ell}{k-\ell} , \quad k = 1,\dots,n \tag{2.5}$$

The following results can be observed from the previous theorem:

1. The limiting zeroes of a pulse transfer function depend critically on the pole excess of the corresponding continuous time system.
2. A continuous time system with a pole excess larger than two will always give a pulse transfer function with zeroes outside the unit disc provided that the sampling period is sufficiently short. This may happen for quite reasonable sampling periods, and sampled data systems with unstable inverses are thus quite common.

An example is given to illustrate the above discussion.

Example 2.1:

Consider a system with transfer function

$$G(s) = \frac{1}{(s+1)^3}$$

The corresponding pulse transfer function can be computed as

$$H(z) = \frac{b_1 z^2 + b_2 z + b_3}{(z-e^{-T})^3}$$

where:

$$b_1 = 1 - (1 + T + T^2/2)e^{-T}$$

$$b_2 = (-2 + T + T^2/2)e^{-T} + (2 + T - T^2/2)e^{-2T}$$

$$b_3 = (1 - T + T^2/2)e^{-2T} - e^{-3T}$$

H(z) has a zero outside the unit disc if $0 < T < 1.8399$.

The result of Theorem 1 can be understood by studying the behavior of the continuous time transfer function $G(s) = s^{-n}$ and its pulse transfer function. The reason why Theorem 1 holds is because for a sampling period T small every system behaves like $G(s) = 1/s^{n-m}$ with the number of poles n and number of zeroes m.

The pulse transfer function corresponding to $G(s) = s^{-n}$ is given by

$$H(z) = \frac{T^n B_n(z)}{n!\ (z-1)^n} \tag{2.6}$$

where $B_n(z)$ is given in Equation (2.4).

The polynomials B_n are found for some values of n using the program given in Appendix A.

$$B_1(z) = 1$$

$$B_2(z) = z + 1$$

$$B_3(z) = z^2 + 4z + 1$$

$$B_4(z) = z^3 + 11z^2 + 11z + 1$$

$$B_5(z) = z^4 + 26z^3 + 66z^2 + 26z + 1$$

The polynomials B_n have zeroes outside or on the unit circle for $n \geq 2$. The unstable zeroes are given in Table 2.1.

TABLE 2.1

UNSTABLE ZEROES OF $B_n(z)$

n	Unstable Zeroes of $B_n(z)$
2	-1
3	-3.732
4	-1, -9.899
5	-2.322, -23.20

Therefore, it can be noticed that there are continuous time systems with stable zeroes such that the corresponding pulse transfer function has unstable zeroes.

It is possible to give a complete characterization of the zeroes of the pulse transfer function for small sampling periods. A similar result for large sampling periods is given by Theorem 2.

Theorem 2:

Let G(s) be a strictly proper rational transfer function with $G(0) \neq 0$ and $\mathrm{Re}P_i < 0$. Then all zeroes of the pulse transfer function go to zero as the sampling period T goes to infinity.

A lower limit on T for stable zeroes was obtained in [Ref. 14], here stated only for simple poles.

$$T > \frac{1}{\delta} \ln[2A(n+1)] \tag{2.7}$$

where:

$$\delta = -\max_i \operatorname{Re} P_i > 0 \tag{2.8}$$

and

$$A = \max_i |A_i / G(0)| \tag{2.9}$$

If G(0) = 0 it follows from Equation (2.10);

$$H(z) = G(0)z^{-1} + (1-z^{-1}) \sum_{i=1}^{n} A_i \frac{e^{P_i T}}{z - e^{P_i T}} \tag{2.10}$$

The corresponding pulse transfer function has one zero at z = 1. The behavior of the rest of the zeroes may be more complex as is shown by Example 2.2.

Example 2.2:

Let continuous time transfer function be

$$G(s) = \frac{s}{[(s+1)^2 + 1](s+2)}$$

Then the corresponding pulse transfer function is

$$H(z) = \frac{e^{-T}(z-1)\{(e^{-T}+\sin T - \cos T)z + e^{-T}[1-e^{-T}(\sin T+\cos T)]\}}{2z(z-e^{-2T})(z^2 - 2ze^{-T}\cos T + e^{-2T})}$$

The zeroes are $z_1 = 1$ and

$$z_2 = \frac{e^{-2T}(\sin T+\cos T) - e^{-T}}{e^{-T} + \sin T - \cos T}$$

For control system design in discrete time domain it is important to know whether the zeroes of pulse transfer function are inside the unit disc or not. When all zeroes are inside the unit disc the sampled system has a stable inverse and all its zeroes can be cancelled. It is therefore of interest to find sufficient conditions which guarantee that all zeroes of a sampled transfer function are inside the unit disc. The criteria are given in Theorem 3 by [Ref. 6].

Theorem 3:

If G(s) is a strictly proper, rational transfer function with

(i) $\mathrm{Re}P_i < 0$

(ii) $G(s) = 0$

(iii) $-\pi < \arg G(i\omega) < 0, \quad \text{for} \quad 0 < \omega < \infty$

Then all the zeroes of the corresponding pulse transfer function H(z) are stable, so that criteria for no unstable zeroes are given both in terms of G(s) and in terms of conditions on the Nyquist curve $G(i\omega)$.

B. FREQUENCY DOMAIN ANALYSIS OF THE ZEROES OF SAMPLED DATA SYSTEM

In this section, the number of zeroes outside the unit disc and the low frequency characteristic of the pulse transfer function are analyzed from the transfer function of a set of continuous time processes.

The discrete frequency response of a continuous process $G^*(\omega)$ can be derived from Equation (2.11) by substituting $z = \exp(j\omega T)$:

$$G(z) = K \frac{(z-\sigma_1) \ldots (z-\sigma_{n-1})}{(z-p_1) \ldots (z-p_n)} \tag{2.11}$$

or it can be expressed by the holding device transfer function $H(\omega)$ and process transfer function $G(\omega)$. Then it becomes:

$$G^*(\omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} H(\omega+k\Omega)\, G(\omega+k\Omega) \tag{2.12}$$

where k is an integer and $\Omega = 2\pi/T$. $G(\omega)$ is the frequency response of the continuous time system $G(s)$.

For zero-order hold, the holding device frequency response is

$$H(\omega+k\Omega) = \frac{1 - e^{-j\omega T}}{j(\omega + k\Omega)} \tag{2.13}$$

Substituting Equation (2.13) into Equation (2.12) and taking into account that $G(-\omega) = \hat{G}(\omega)$, where $\hat{G}$ is the conjugated value of G, we obtain

$$G^*(\omega) = \frac{1 - e^{-j\omega T}}{j\omega T} G(\omega)[1 - f(\omega) + \varepsilon(\omega)] \qquad (2.14)$$

where:

$$f(\omega) = \frac{\omega}{\Omega - \omega} \frac{G(\Omega-\omega)}{G(\omega)} \qquad (2.15)$$

$$\varepsilon(\omega) = \frac{\omega}{G(\omega)} \sum_{r=1}^{\infty} \left\{\frac{G(r\Omega + \omega)}{r\Omega + \omega} \cdot \frac{G((r+1)\Omega - \omega)}{(r+1)\Omega - \omega}\right\} \underset{-}{\sim} 0 \qquad (2.16)$$

and r is an integer.

The number of the zeroes outside the unit disc are given by Theorem 4 [Ref. 10].

Theorem 4:

If a continuous process satisfies Equation (2.16) and the condition

$$\left|\frac{\hat{G}(\Omega-\omega)}{G(\omega)}\right| \leq 1 \qquad (2.17)$$

the number of poles of G(z) outside the unit disc is zero and the phase angle $\phi(\Omega/2) = \text{arc } G(\Omega/2)$ is between 0 and $-\pi$ then its pulse transfer function possesses no zeroes outside the unit disc. If the value of phase angle is between $-\pi$ and -2π one of the zeroes lies outside the unit disc, etc.

$$G^*(z) = K^* \frac{(z-\sigma_1) \ldots (z-\sigma_4)}{(Z-e^{-T/T_1}) \ldots (z-e^{T/T_5})}$$

Thus, inverse stable continuous processes frequently possess inverse unstable pulse transfer functions. For instance if in a transfer function satisfying the conditions given by Equations (2.16) and (2.17), $\tau_i > T$ and $T_i > T$ are real and positive for every i and the orders of its denominator and numerator differ by more than two, then usually the pulse transfer function is inversely unstable.

Also, from Theorem 4, the phase of the continuous frequency response at the Nyquist frequency $\Omega/2$ is directly related to the number of unstable discrete time zeroes. High frequency dynamics (where high frequency is intended as above the Nyquist frequency) influence the phase at the Nyquist frequency, so that it can cause the number of unstable discrete time zeroes to increase. This high frequency pole may come from the structure of the measuring devices, actuators or other hardware parts of the physical realization. Usually, we neglect these terms in the transfer function. The following example can give an idea about this problem.

Example 2.3:

Let the transfer function of the plant be

$$H(s) = \frac{1}{s^2 - 1}$$

which has a pulse transfer function given by

$$H(z) = \frac{.54308z + .54307}{z^2 - 3.08616z + 1}$$

with one zero at z = -0.998.

By adding the extra dynamics D(s) as

$$D(s) = \frac{w_n^2}{s^2 + 2\zeta w_n + w_n^2}$$

the zeroes of the pulse transfer function of the entire system for $\zeta = 0.9$ and some w_n values are given in Table 2.2 and plotted in Figure 2.1.

TABLE 2.2

ZEROES LOCATIONS FOR DIFFERENT w_n VALUES

W_n	Zeroes		
7	-3.162004	-0.01349725	-0.1775817
8	-2.866706	-0.009074569	-0.1386477
9	-2.633762	-0.005823303	-0.1097788
10	-2.447441	-0.003552481	-0.08844686
11	-2.296317	-0.002030142	-0.07263207
12	-2.172057	-0.001109288	-0.06071956
13	-2.068539	-0.0004927621	-0.05170227
14	-1.981240	-0.0002742233	-0.04454243
15	-1.906793	-0.00009020962	-0.03895098
16	-1.842641	-0.00006002390	-0.03435726
17	-1.786843	-0.00004560289	-0.03068041
18	-1.737905	-0.00003143626	-0.02745582
19	-1.694647	-0.00005455861	-0.02487628
20	-1.656146	-0.0001198984	-0.02266212

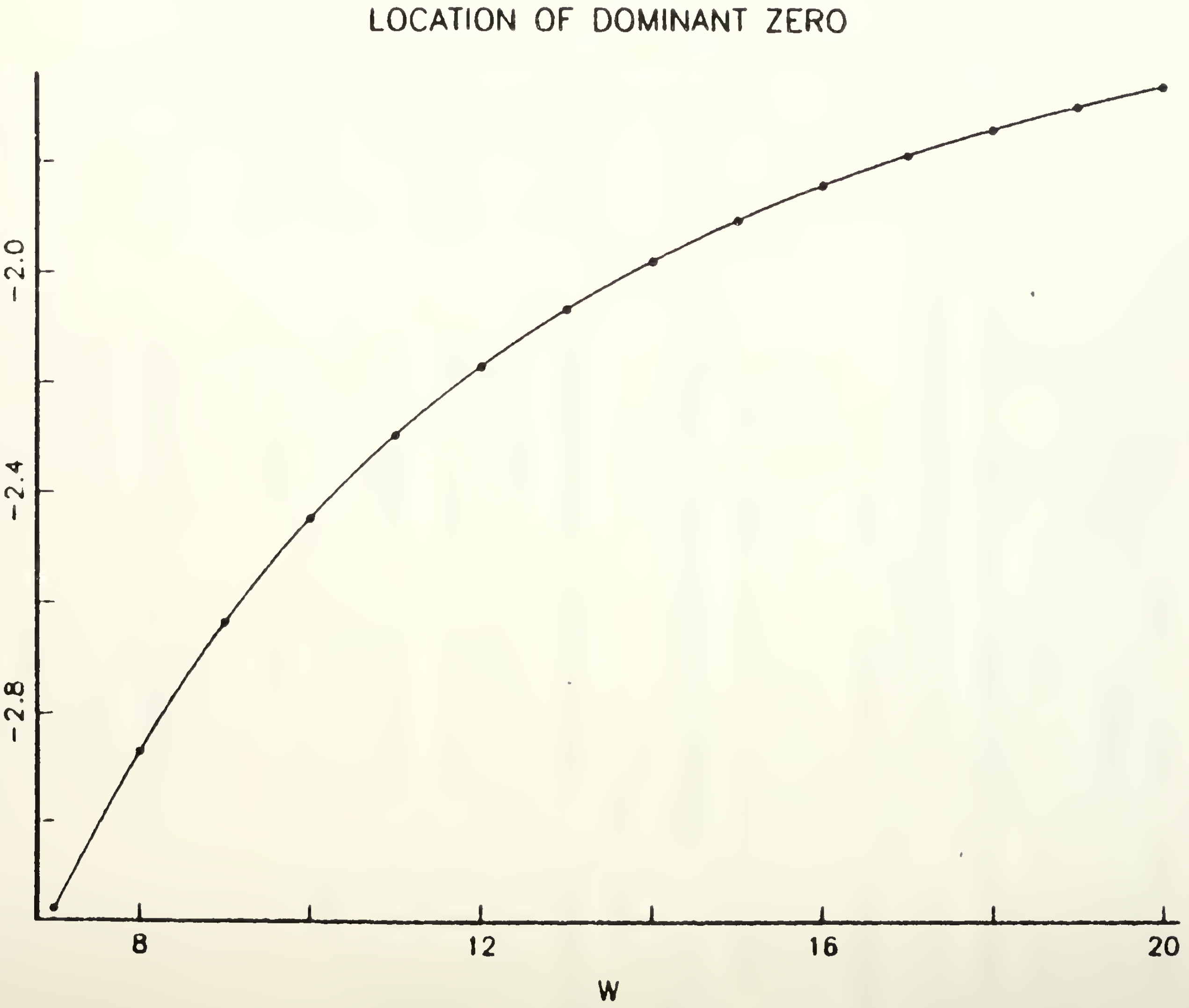

Figure 2.1. Dominant Zero Location vs. W.

We notice that as w_n decreases, the unstable zero gets more and more unstable.

III. INDIRECT ADAPTIVE CONTROL FOR DISCRETE TIME SYSTEMS

As mentioned in the Introduction, the main advantage of indirect adaptive control systems comes from the fact that it is applicable to discrete time systems with unstable zeroes. It is assumed that the system is as given in Figure 3.1,

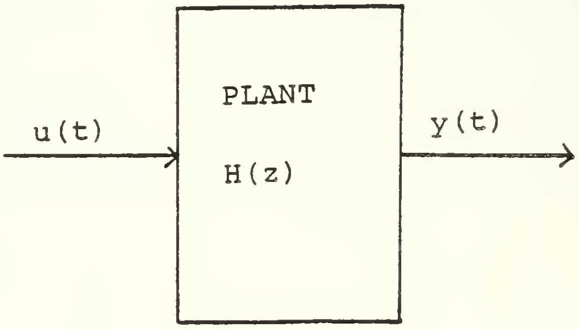

Figure 3.1. Single Input--Single Output Plant System

The pulse transfer function is given by

$$H(z) = \frac{r(z)}{p(z)} \tag{3.1}$$

r(z) and p(z) being polynomials given by:

$$p(z) = z^n + p_1 z^{n-1} + \ldots + p_n \tag{3.2}$$

$$r(z) = r_1 z^{n-1} + r_2 z^{n-2} + \ldots + r_n \tag{3.3}$$

The degree of the denominator polynomial p(z) determines the system order n. Also, if we assume the plant to be causal, the polynomial r(z) has degree at most n-1. The sequences u(t) and y(t) denote the system input and output, respectively. An alternative way of describing the plant is by its difference equation written in operator form, as

$$p(D)y(t) = r(D)u(t) \tag{3.4}$$

where:

$$p(D) = 1 + p_1 D + \ldots + p_n D^n \tag{3.5}$$

$$r(D) = r_1 D + \ldots + r_n D^n \tag{3.6}$$

and $D = q^{-1}$ is the backward-shift operator, defined as $Dy(t) = y(t-1)$.

In the adaptive control problem the parameters in p(D) and r(D) are assumed to be unknown. Only the system order n is assumed to be known to the designer. Hence, two problems appear at this point:

1. Parameter estimation; and
2. Compensator structure.

The general block diagram of the indirect adaptive control problem is given in Figure 3.2. Estimation of the parameters of the plant is mentioned in Chapter III.C. The compensator structure is derived from the pole placement problem for

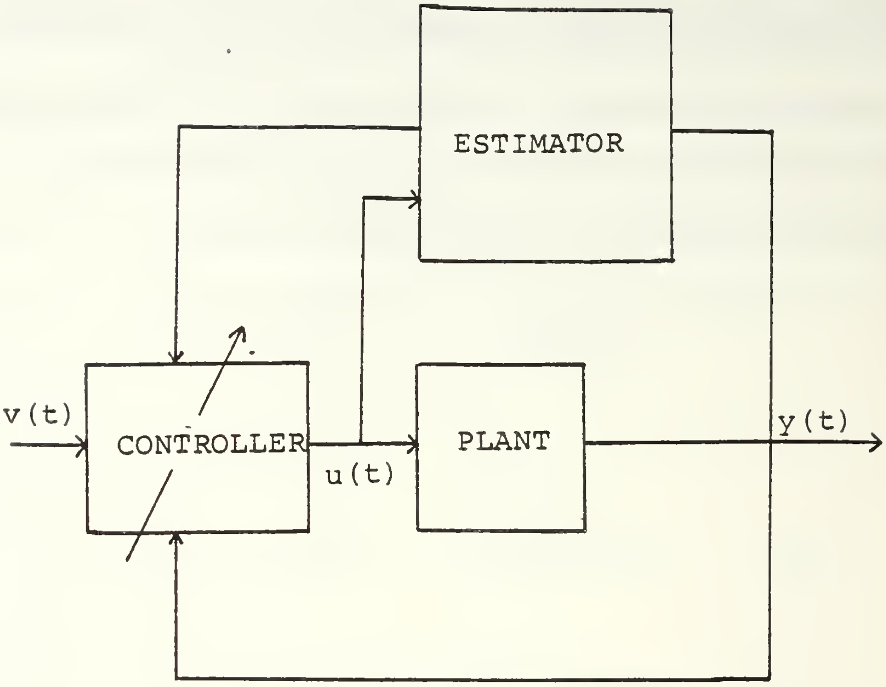

Figure 3.2. General Block Diagram of Indirect Adaptive Control

linear systems; and it is computed on the basis of the estimated plant parameters.

In the next section, we discuss the pole-placement problem which will be the basis for the determination of a suitable controller structure.

A. POLE PLACEMENT DESIGN BASED ON INPUT-OUTPUT MODELS

The purpose of pole-placement by state-feedback is to determine a feedback controller so that all poles of the closed-loop system assume prescribed values. This can be easily achieved if all state variables of the plant are measured. In general, the states may not be directly available. However, under certain observability conditions the state can be observed on the basis of input-output measurements. For this reason observers are used to estimate the state of any observable realization. Along these lines, the controller can be considered as composed of an observer and a gain matrix.

Consider a state-space representation of the plant assumed to be controllable and observable:

$$\underline{x}(t+1) = \Phi\underline{x}(t) + \Gamma u(t) \tag{3.7}$$

$$y(t) = C\underline{x}(t) \tag{3.8}$$

with Φ, Γ, C matrices of dimensions $n \times n$, $n \times 1$ and $1 \times n$, respectively.

The block diagram of the controlled system combined with the observer-controller is given in Figure 3.3. $V(t)$ is an external input which will be discussed in Chapter IV. If we restrict ourselves to the class of linear control inputs we can write:

$$u(t) = -L\hat{x}(t) + v(t) \tag{3.9}$$

as discussed in [Ref. 3], where L is a constant matrix as

$$L = [l_1\ l_2\ \dots\ l_n]$$

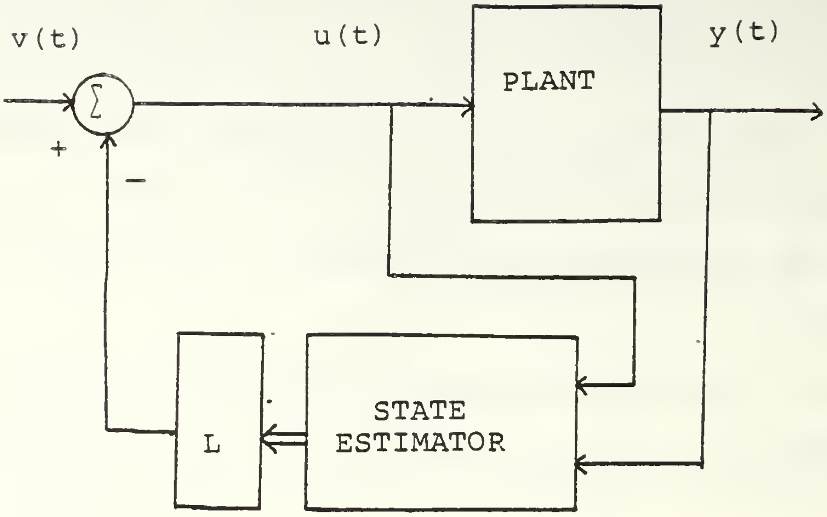

Figure 3.3. Block Diagram of the Controlled System

and $\hat{x}(t)$ indicates the estimated values of the actual states of the plant. It is a well-known result in systems theory [Ref. 3] that we may define the observer dynamics as

$$\underline{\hat{x}}(t+1) = \Phi\underline{\hat{x}}(t) + \Gamma u(t) + K[y(t) - C\underline{\hat{x}}(t)] \tag{3.10}$$

where K is a matrix such that

$$K^T = [k_1 \quad k_2 \quad \ldots \quad k_n]$$

and Φ-KC is a matrix with eigenvalues inside the unit circle.

Rearranging Equation (3.10) we obtain the state-space dynamical equations of the controller as a two input--one output linear system.

$$\underline{\hat{x}}(t+1) = [\Phi-KC]\underline{\hat{x}}(t) + \Gamma u(t) + Ky(t) \tag{3.11}$$

$$u(t) = -L\hat{x}(t) + v(t) \tag{3.12}$$

To convert the state-space representation of the pole-placement problem to a transfer function (using polynomials) form, taking the z transform of Equations (3.11) and (3.12) we obtain

$$\hat{X}(z) = (zI-Q)^{-1}\Gamma U(z) + (zI-Q)^{-1}KY(z) \tag{3.13}$$

$$U(z) = -L(zI-Q)^{-1}\Gamma U(z) - L(zI-Q)^{-1}KY(z) + V(z) \tag{3.14}$$

where we define the matrix $Q = \Phi-KC$. Also we can define the rational functions

$$\begin{aligned} -L(zI-Q)^{-1}\Gamma &= \frac{L\,\mathrm{adj}\,(zI-Q)\Gamma}{\det(zI-Q)} \\ &= \frac{k(z)}{q(Z)} \end{aligned} \tag{3.15}$$

and

$$\begin{aligned} -L(zI-Q)^{-1}K &= \frac{-L\,\mathrm{adj}\,(zI-Q)K}{\det(zI-Q)} \\ &= \frac{h(z)}{q(z)} \end{aligned} \tag{3.16}$$

where q(z), the observer polynomial, is an arbitrary stable polynomial. Arbitrariness of q(z) is guaranteed by the assumption of the plant being observable, while h(z) and k(z), the controller polynomials, have to be computed, on the basis of the plant dynamics and q(z).

Thus, the structure of the control input u can be described as:

$$U(z) = \frac{k(z)}{q(z)} U(z) + \frac{h(z)}{q(z)} Y(z) + V(z) \tag{3.17}$$

or it may be expressed in a difference operator form as

$$q(D)u(t) = K(D)u(t) + h(D)y(t) + q(D)v(t) \tag{3.18}$$

To make the notation more attractive, let us write Equation (3.18) as

$$u(t) = \frac{k(D)}{q(D)} u(t) + \frac{h(D)}{q(D)} y(t) + v(t) \tag{3.19}$$

The block diagram of the above controlled system is in Figure 3.4. Hence, the closed-loop system defined from external input v(t) to the output y(t) has transfer function

$$y(t) = \frac{q(D)r(D)}{p(D)[q(D)-k(D)] - r(D)h(D)} v(t) \tag{3.20}$$

In the pole-placement problem, the goal is to determine the compensator parameters (k,h,q) so that the closed-loop poles are as we desire,

$$y(t) = \frac{r(D)}{p^*(D)} v(t) \tag{3.21}$$

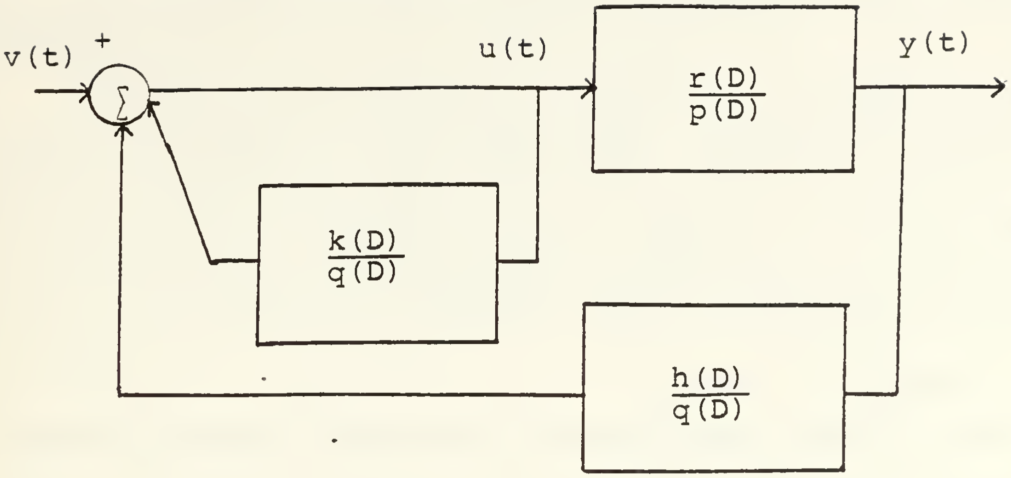

Figure 3.4. Transfer Function Form of the Closed-Loop System

$p^*(D)$ being an arbitrary stable polynomial in the difference operator D. Roots of $p^*(D)$ are the new assigned poles of the closed-loop system.

By equating Equations (3.20) and (3.21), we obtain that k, h, q have to satisfy the polynomial equation

$$p(D)[q(D) - k(D)] - r(D)h(D) = q(D)p^*(D) \tag{3.22}$$

which can be written as

$$k(D)p(D) + h(D)r(D) = F(D) \tag{3.23}$$

where:

$$F(D) = q(D)[p(D) - p^*(D)].$$

Therefore, the problem of determining a suitable compensator for pole-placement is equivalent to solving the polynomial equation (3.23) which is known by the name of Diophantine equation. The conditions by which this equation can be solved, together with the method of solution itself, are given in the following section.

B. DIOPHANTINE EQUATION

The general form of the Diophantine equation in the unknown polynomials K(z) and H(z) is given by:

$$F(z) = k(z)P(z) + h(z)R(z) \tag{3.24}$$

with:

$$k(z) = k_o + k_1 z + \ldots + k_m z^m, \quad k_m \neq 0 \tag{3.25}$$

$$h(z) = h_o + h_1 z + \ldots + h_m z^m \tag{3.26}$$

$$P(z) = P_o + p_1 z + \ldots + p_n z^n \tag{3.27}$$

$$R(z) = r_o + r_1 z + \ldots + r_n z^n \tag{3.28}$$

and

$$F(z) = f_o + f_1 z + f_2 z^2 + \ldots + f_{n+m} z^{n+m} \tag{3.29}$$

where:

k_i, h_i, p_i, r_i and f_i are constant, not necessarily all nonzero.

In the general pole placement setting, the polynomials p(z), r(z), F(z) are given, while h(z), k(z) are unknown.

In the rest of this section, we determine the conditions by which the Diophantine equation can be solved and the method of solution.

By substituting Equations (3.25)-(3.29) into (3.24), we obtain:

$$f_o + f_1 z + \ldots + f_{n+m} z^{n+m} = (p_o + p_1 z + \ldots + p_n z^n)(k_o + k_1 z + \ldots + k_m z^m) + (r_o + r_1 z + \ldots + r_n z^n)(h_o + h_1 z + \ldots h_m z^m) \qquad (3.30)$$

and equating the coefficients of the same power of z yields the linear relation:

$$C^T S_m = [f_o \quad f_1 \quad f_2 \quad \ldots \quad f_{n+m}] \qquad (3.31)$$

where the vector C is defined as:

$$C = [k_o \quad h_o \quad k_1 \quad h_1 \quad \ldots \quad k_m \quad h_m]^T \qquad (3.32)$$

and the matrix S_m as:

$$S_m = \begin{bmatrix} p_o & p_1 \cdots p_{n-1} & p_n & 0 & 0 & \cdots & 0 \\ r_o & r_1 \cdots r_{n-1} & r_n & 0 & 0 & \cdots & 0 \\ 0 & p_o \cdots p_{n-2} & p_{n-1} & p_n & 0 & \cdots & 0 \\ 0 & r_o \cdots r_{n-2} & r_{n-1} & r_n & 0 & \cdots & 0 \\ & \vdots & & & & \vdots & \\ 0 & 0 \cdots 0 & & p_0 & p_1 & \cdots & p_n \\ 0 & 0 \cdots 0 & & r_0 & r_1 & \cdots & r_n \end{bmatrix} \qquad (3.33)$$

Equation (3.31) is a linear algebraic equation in the unknown vector C. The matrix S_m consists of m+1 block rows; each block row has two rows and can be obtained by shifting its previous block row to the right by one column. It is a $2(m+1) \times (n+m+1)$ matrix.

In order for a solution to exist, the matrix S_m has to be full column rank [Ref. 5]. This can be satisfied if $2(m+1) \geq n+m+1$, or $m \geq n-1$.

When S_m is a square matrix ($m = n-1$), it is called the Sylvester matrix of P and R, which has nonzero determinant provided polynomials P and R are mutually coprime, i.e., they do not have common factors [Ref. 6].

We can summarize the steps to find h and k, mentioned above, as:

1. Let n be the order of the system.
2. Choose q(z) to be an arbitrary polynomial of degree n.

3. Form the matrix S_m using P(z) and R(z).

4. Set up the polynomial F(z) as:

$$F(z) = q(z)[p(z) - p^*(z)].$$

5. Solve for parameters of h(z) and k(z).

6. Set up the polynomials h(z) and k(z), or h(D) and k(D) using the relation $D = z^{-1}$.

C. PARAMETER ESTIMATION

As mentioned above, in indirect adaptive control, we have to estimate the parameters of the plant, where the parameters are the coefficients of the transfer function

$$H(z) = \frac{r(z)}{p(z)} \qquad (3.34)$$

We can compute the controller parameters h(D) and k(D) from the Diophantine equation on the basis of the estimated plant (say r,p).

A large number of different identification methods are available. In the literature, one broad distinction is between on-line methods and off-line methods. In the off-line case, it is presumed that all data are available prior to analysis. Consequently, the data may be treated as a complete block of information, with no strict time limit on the process of analysis. In contrast to the off-line case, the on-line case deals with sequential data, which requires the paraemter estimates to be recursively updated within the time limit imposed by the sampling period.

As mentioned, on-line estimation schemes produce an updated parameter estimate within the time span between successive samples.

Also, the on-line methods are the only alternative if the estimation is going to be used in an adaptive controller or if the process is time-varying. In this thesis, we consider two classes of estimation algorithms:

1. projection;
2. recursive least-squares.

Before proceeding, it can be said that input-output characteristics of a wide-class of linear and nonlinear deterministic dynamical systems can be described by a model expressed in the following form:

$$y(t) = \Phi(t-1)^T\theta_o \tag{3.35}$$

where $y(t)$ denotes the system output at time t, $\Phi(t-1)$ denotes a vector given as:

$$\Phi(t-1)^T = [y(t-1)\ y(t-2)\ \ldots\ y(t-n)\ u(t-1)\ \ldots\ u(t-n)] \tag{3.36}$$

and θ denotes the parameters of the plant, described as:

$$\theta_o = \{-p_1,-p_2,\ldots,p_n,r_1,r_2,\ldots,r_m\}^T \tag{3.37}$$

The following example illustrates the representation of the plant as given in Equation (3.35).

Example 3.1:

Suppose the pulse transfer function is

$$H(z) = \frac{z+1}{z^2+2z+3}$$

It is also expressed in the following form:

$$H(z^{-1}) = \frac{z^{-1}+z^{-2}}{1+2z^{-1}+3z^{-2}}$$

$$H(q^{-1}) = \frac{q^{-1}+q^{-2}}{1+2q^{-1}+3q^{-2}}$$

Hence, the difference equation becomes

$$y(t) = -2y(t-1) - 3y(t-2) + u(t-1) + u(t-2)$$

which can be expressed as:

$$y(t) = [y(t-1)\ \ y(t-2)\ \ u(t-1)\ \ u(t-2)] \begin{bmatrix} -2 \\ -3 \\ 1 \\ 1 \end{bmatrix}$$

The rest of this section illustrates the estimation methods. The projection algorithm and the recursive least-squares algorithm are analyzed sequentially.

(I) Projection Algorithm

By the Projection Algorithm, the sequence of estimates $\hat{\theta}(t)$ is recursively computed as:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{a\Phi(t-1)}{c + \Phi(t-1)^T\Phi(t-1)}[y(t)-\Phi(t-1)^T\hat{\theta}(t-1)] \quad (3.38)$$

with arbitrary initial estimate $\hat{\theta}(0)$ and $c > 0$; $0 < a < 2$. The detailed derivation steps are given by [Ref. 2].

This algorithm is also known as the normalized least-mean-squares (NLMS) algorithm. The algorithm results from the following optimization problem: Given $\hat{\theta}(t-1)$ and y(t0, determine $\hat{\theta}(t)$ so that

$$J = \frac{1}{2}||\hat{\theta}(t) - \hat{\theta}(t-1)||^2 \quad (3.39)$$

is minimized subject to

$$y(t) = \Phi(t-1)^T\hat{\theta}(t) \quad (3.40)$$

The main properties of the projection algorithm are the following:

1. Estimation of θ at time t, $\hat{\theta}(t)$ is always closer to the actual value of θ than the preceding estimated value $\hat{\theta}(t-1)$.

$$||\hat{\theta}(t) - \theta_o|| < ||\hat{\theta}(t-1) - \theta_o|| < ||\hat{\theta}(0) - \theta_o||; \quad (3.41)$$

$$t \geq 1$$

2. When the time goes to infinity, the error between the actual output of the system and its predicted value will converge to zero.

$$\lim_{t\to\infty} \frac{e(t)}{[c + \Phi(t-1)^T \Phi(t-1)]^{1/2}} = 0 \qquad (3.42)$$

where:

$$e(t) = y(t) - \Phi(t-1)^T \hat{\theta}(t-1)$$

3. $\lim_{t\to\infty} ||\hat{\theta}(t) - \hat{\theta}(t-k)|| = 0$ for any finite k (3.43)

The adaptation rate converges to zero as t → ∞.

(II) Recursive Least Squares Algorithm

According to Gauss the principle on which the least square estimates are based is that the unknown parameters of the process should be chosen in such a way that the sum of the squares of the differences between the actually observed and computed values multiplied by numbers that measure the degree of precision is a minimum.

The recursive least squares algorithm is given by the following equation in [Ref. 2].

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{P(t-2)\Phi(t-1)[y(t)-\Phi(t-1)^T\hat{\theta}(t-1)]}{1 + \Phi(t-1)^T P(t-2)\Phi(t-1)} \qquad (3.44)$$

for t $\geq$ 1 and

$$P(t-1) = P(t-2) \frac{P(t-2)\Phi(t-1)\Phi(t-1)^T P(t-2)}{1+\Phi(t-1)^T P(t-2)\Phi(t-1)} \qquad (3.45)$$

with $\hat{\theta}(0)$ given and $P(-1)$ is any positive definite matrix P_o.

The algorithm results from the minimization of the following quadratic cost function:

$$J_N(\theta) = \frac{1}{2} \sum_{t=1}^{N} \{y(t)-\Phi(t-1)^T\theta\}^2 + \frac{1}{2}(\theta-\hat{\theta}(0))^T P_o^{-1}(\theta-\hat{\theta}(0)) \qquad (3.46)$$

We can observe that the cost function represents the sum of squares of the output prediction error e(t).

$$e(t) = y(t) - \Phi(t-1)^T\hat{\theta} \qquad (3.47)$$

The second term of the right-hand side of the cost function accounts for the initial parameter estimates weighted by the matrix P_o. In this way, we can consider P_o (the initial condition of P(t) in Equation (3.45)) as the "confidence" on the initial conditions $\hat{\theta}(0)$.

The least squares algorithm, as can be seen from the simulation results, has much faster convergence than the projection algorithm.

D. PERSISTENCY OF EXCITATION

In order to guarantee global stability of indirect adaptive control algorithms, the input to the plant has to be persistently exciting, which in turn implies global convergence of the plant parameters to the corresponding actual values [Ref. 3]. To obtain consistent estimates of the plant parameters, it is necessary that the input signal to the plant be sufficiently rich in frequencies, and excites all modes of the plant.

In a closed loop set-up, the control input is the sum of an external input signal v(t) and a feedback signal from the adaptive controller.

The feedback signal may in principle cancel any excitation contained in the external input signal. This problem and the potential for unbounded growth of the control and output signals have made the guarantee of persistent excitation a difficult problem. Recently, Elliot [Ref. 8] gave sufficient conditions which guarantee persistency of excitation. The following theorems summarize these conditions.

Theorem 4.1:

Let w be the number of parameters estimated. In order to guarantee global convergence of the plant parameters to their true values, the following conditions have to be satisfied:

1. The external input v(t) should consist of a sum of 2w sinusoids.
2. The compensator parameters should be updated each N samples, with $N \geq 10n$.

Therefore, in this thesis report, block processing is used in the sense that N data samples are taken, and N iterations of the recursive least-squares algorithm are performed between control parameters $(\hat{h},\hat{k})$ updates. To avoid time variation of the plant dynamics during the spanning process, the control parameters are held constant during spanning blocks (of length N) and are changed only between them.

As we can see from Theorem 4.1, these two conditions are sufficient to guarantee parameter convergence for any initial conditions as:

$$\lim_{k\to\infty} \hat{\theta}_k = \theta^*$$

$\hat{\theta}_k$ being the estimate of θ^* at time t_k.

In the next section, we will examine finite time persistency of excitation.

E. FINITE TIME PERSISTENCY OF EXCITATION

It is clear that the persistency of excitation condition on the external input v(t) is the main limitation about this adaptive control algorithm. Recently, in a report by Cristi [Ref. 15], a possible solution to this problem has been given, by stopping parameter adaptation when the performances are close to the desired ones. More precisely, the model output error between desired output of the closed-loop system and controlled plant's output is the measure of how far the system is from the desired performance (i.e., pole placement) and adaptation is stopped whenever the error falls below an arbitrary, preassigned threshold. The configuration of the system is given in Figure 3.5.

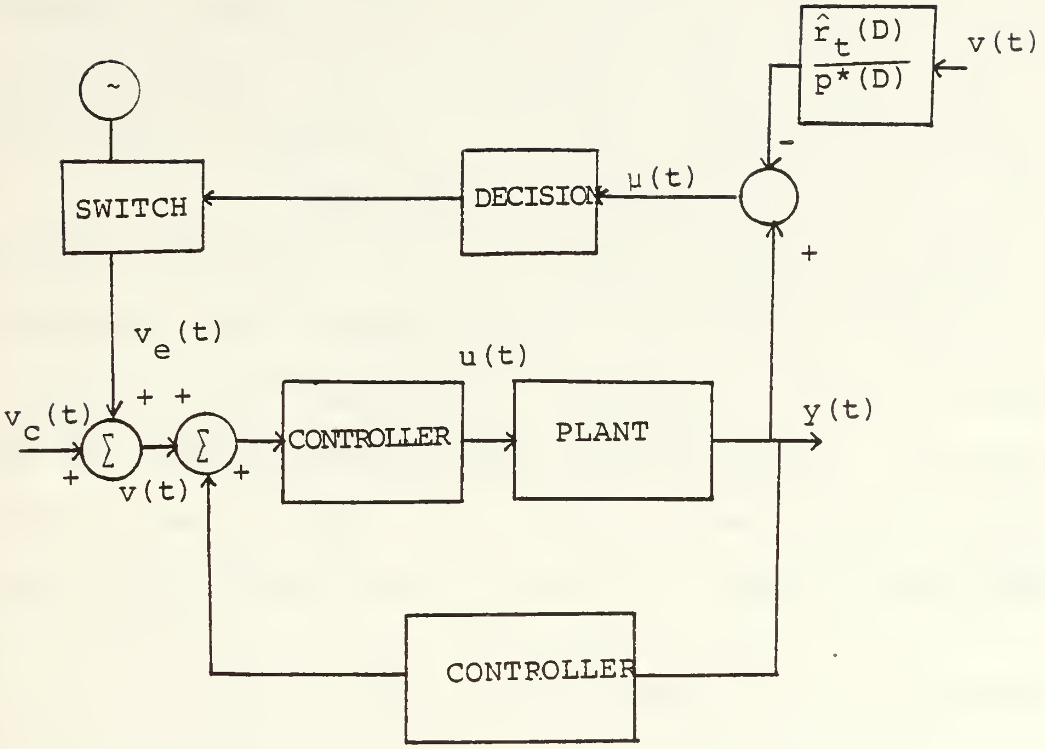

Figure 3.5. Indirect Adaptive Control Scheme with Added Finite Time P.E.

The model output error can be defined as

$$\mu(t) = y(t) - \frac{\hat{r}_t(D)}{p^*(D)} v(t) \tag{3.48}$$

The finite time persistency of excitation algorithm can be given as follows:

1. $t = t+1$.
2. Compute $\mu(t)$ from Equation (3.48).
3. If $|\mu(t)| > \varepsilon$, compute compensator parameters as in Chapter III.
4. If $|\mu(t)| \leq \varepsilon$ go to 1.

A major difficulty with this algorithm is to be able to guarantee global stability of the whole system. Therefore we have to prove that the signals in the loop are bounded, provided the external input v(t) is bounded. Global stability of the closed-loop system is proved below.

When time goes to infinity and persistency of excitation is present, parameters of the estimates of the plant polynomials $\hat{r}_t$ and $\hat{P}_t$ converge to the actual values r and p. Therefore, the difference between measured output and desired output tends to zero, and also $\mu(t)$ tends to zero. Thus, by definition of limit, there exists an instant t_o such that

$$|\mu(t)| < \varepsilon$$

for all $t > t_o$ and for any given $\varepsilon > 0$. Rearranging Equation (3.48) yields

$$y(t) = \frac{\hat{r}(D)}{p^*(D)} v(t) + \mu(t) \qquad (3.49)$$

which can be expressed on a block diagram form as in Figure 3.6.

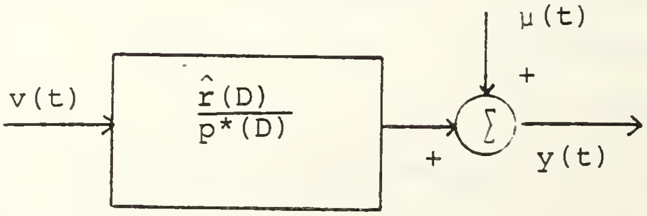

Figure 3.6. Block Diagram Representation of Equation (3.49)

It can be interpreted as a single input single output linear stable process with bounded input $v(t)$ and bounded disturbance $\mu(t)$. From the fact that stable linear systems with a bounded input produce a bounded output, the global stability results follow easily. So that with finite time persistency excitation, the closed-loop system is eventually equivalent to a system with poles as desired, bounded zeroes with a bounded output disturbance $\mu(t)$.

We can generate the external input $v(t)$ as

$$v(t) = v_e(t) + v_c(t) \tag{3.50}$$

with $v_c(t)$ the external desired command and $v_e(t)$ an added persistency excitation signal. When the adaptation is continuing, the external input is composed of $v_c(t)$ and $v_e(t)$, but after stopping the adaptation persistency of excitation is not needed, it will be identical to the longer, so that $v_c(t)$ can decay to zero.

In this way, the adaptive algorithm is activated only when the performance index $\mu(t)$ is larger than a minimum threshold. In the next chapter, simulation studies will show their efficiency via some examples.

IV. SIMULATION STUDIES

This research report includes three computer programs. The program named CONDIS, given in Appendix B, is used to investigate the behavior of the zero in sampled data systems. This program computes the parameters of the discrete time transfer function from the parameters of its continuous time, and the sampling rate. This program has been used to investigate the perturbations of the zeroes for different values of the sampling interval.

The other two programs given in Appendix C and D simulate the indirect adaptive control using recursive least squares and projection algorithms. Entering order and numerator, denominator parameters of the plant, observer polynomial and desired closed-loop characteristic polynomial, the program simulates the entire system in an interactive fashion. Also it is capable of graphical and tabulation results.

The adaptive controller presented above has been simulated for several different plant dynamics.

Example 4.1:

Let the discrete time transfer function of the plant be:

$$H(z) = \frac{z + 2}{z^2 - 2z + 0.75} = \frac{z + 2}{(z - 1.5)(z - 0.5)}$$

The plant has one unstable zero, $z = -2$, and two poles $p_1 = 1.5$ and $p_2 = 0.5$.

The polynomials $q(z)$ and $p^*(z)$ are chosen to be stable polynomials with degree n.

In particular, let

$$q(z) = z^2 - 0.3z - 0.28 = (z - 0.7)(z + 0.4)$$

and

$$p^*(z) = z^2 - 1.1z + 0.3 = (z - 0.5)(z - 0.6)$$

both having stable roots, i.e., inside the unit disc in the z-plane. Before going into simulation study, the problem is solved analytically by comparing their responses. By writing the plant dynamics in shift-operator form,

$$H(q^{-1}) = \frac{q^{-1} + 2q^{-2}}{1 - 2q^{-1} + 0.75q^{-2}}$$

we can derive the difference equation of the given system as

$$y(t) = 2y(t-1) - 0.75y(t-2) + u(t-1) + 2u(t-2)$$

$$y(t) = \Phi(t-1)^T\theta^*$$

where

$$\Phi(t-1) = \begin{bmatrix} -y(t-1) \\ -y(t-2) \\ u(t-1) \\ u(t-2) \end{bmatrix}$$

and

$$\theta^{*} = \begin{bmatrix} -2 \\ 0.75 \\ 1 \\ 2 \end{bmatrix}$$

θ^* being the vector of the actual parameters of the plant. Generally, it can be partitioned as:

$$\theta^* = \begin{bmatrix} \underline{p} \\ \underline{r} \end{bmatrix}$$

$\underline{p}$ and $\underline{r}$ being vectors of parameters of the denominator and numerator of the plant, respectively.

From Equation (3.23), considering k(z) and h(z) as controller polynomials, we can write

$$k(z)p(z) + h(z)r(z) = q(z)[p(z) - p^*(z)]$$

In this example we have chosen:

$$p(z) = z^2 - 2z + 0.75$$

$$r(z) = z + 2$$

$$q(z) = z^2 - 0.3z - 0.28$$

$$p^*(z) = z^2 - 1.1z + 0.3$$

By the constraints on the polynomials for solvability of the Diophantine equation, we choose k and h to be first order polynomials as:

$$k(z) = k_1 z + k_o$$

$$h(z) = h_1 z + h_o$$

Substituting these polynomials into the Diophantine equation in matrix form yields:

$$\begin{bmatrix} k_o \\ h_o \\ k_1 \\ h_1 \end{bmatrix}^T \begin{bmatrix} 0.75 & -2 & 1 & 0 \\ 2 & 1 & 0 & 0 \\ 0 & 0.75 & -2 & 1 \\ 0 & 2 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -0.126 \\ 0.117 \\ 0.72 \\ -0.9 \end{bmatrix}^T$$

Since the plan does not have any pole-zero cancellation, the determinant of the matrix S_m is not equal to zero. Hence, it is solvable. Solving the above matrix equation, we compute the control parameters $k_1 = -.899$, $k_o = -0.688$, $h_1 = -0.3900$ and $h_o = 0.195$. The corresponding polynomials of the controller are:

$$k(z) = -.899z - 0.688$$

$$h(z) = 0.39z + 0.195$$

This corresponds to the optimal choice of compensator parameters for the desired pole placement. Therefore the compensator given by the difference equation

$$u(t) = \frac{k(D)u(t) + h(D)y(t)}{q(D)} + v(t)$$

yields closed-loop transfer function

$$H(z) = \frac{Y(z)}{V(z)}$$

$$= \frac{q(z)r(z)}{p(z)[q(z) - k(z)] - r(z)h(z)}$$

which becomes:

$$H(z) = \frac{z+2}{z^2 - 1.1z + 0.3}$$

The step response of this system is given in Figure 4.1. The next approach to the problem is simulation of the system using a computer program. It is observed that from the simulations, after the 10n iterations, the estimated parameter will be closer to θ^* and $\tilde{\theta}$ will tend to zero, where

$$\tilde{\theta} = ||\theta^* - \hat{\theta}||$$

This means that $\hat{p}$ and $\hat{r}$ are converging to the actual parameters, and $\hat{h}$ and $\hat{k}$ converge to the actual values for the desired pole-placement. The desired output and controlled system's output are given in Figure 4.2. Also, parameter error and output prediction error are given in Figures 4.3 and 4.4, respectively. After convergence of the prediction error to zero, the persistency of excitation added to the external command is turned off. In this example, we have chosen the step as external command. External input and plant input are in Figures 4.5 and 4.6, respectively.

If we compare Figures 4.1 and 4.2, analytical and simulated system's results are almost the same.

In the next example, it is considered that one of the plant parameter is changed after some period of time. It is investigated how the system behavior will change in this condition.

Example 4.2:

Let the plant be the same as the one in the previous example. After two blocks length, the zero of the plant is

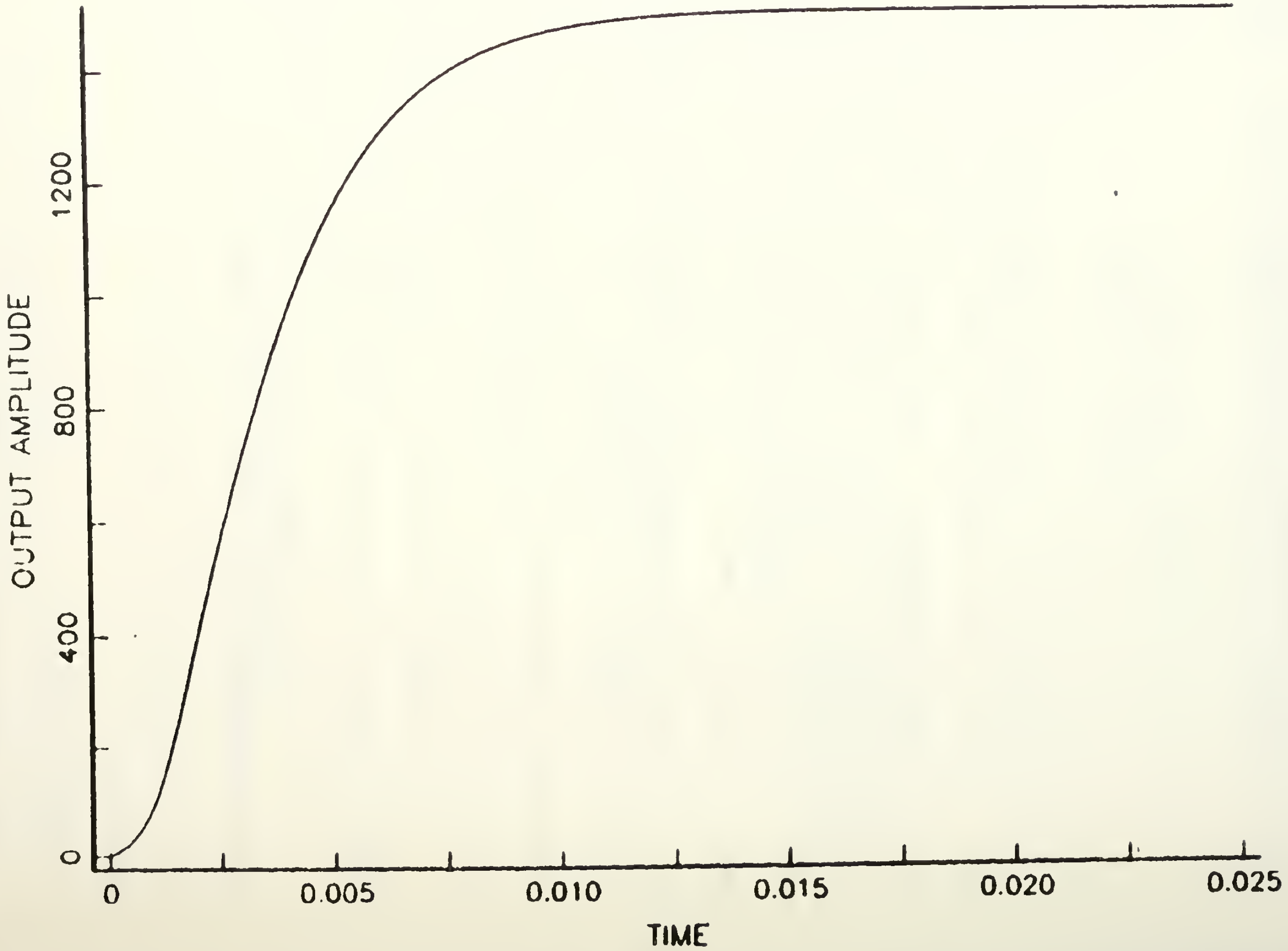

Figure 4.1. Step Response of the Closed-Loop System (Example 1)

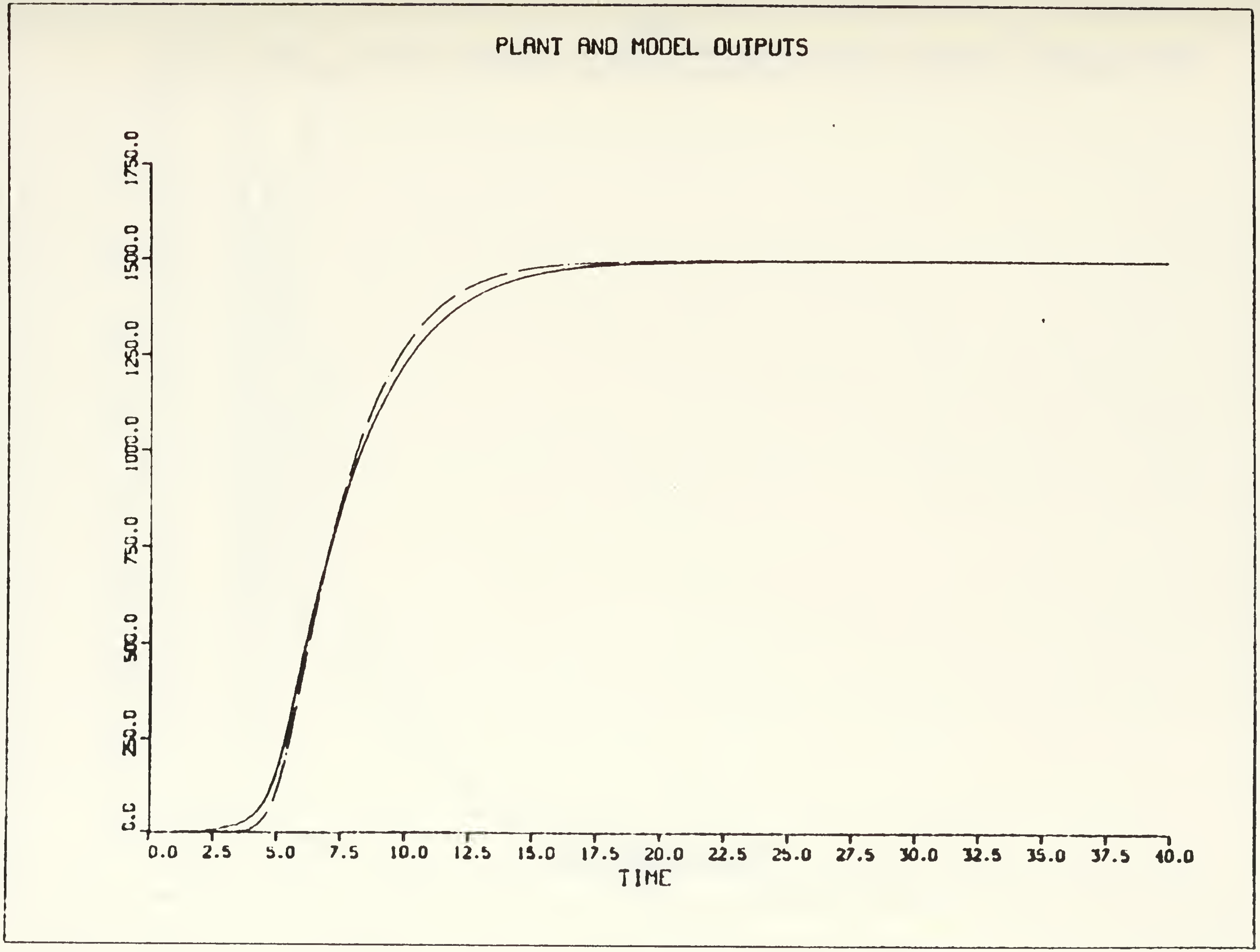

Figure 4.2. Plant and Model Outputs vs. Time (Example 1)

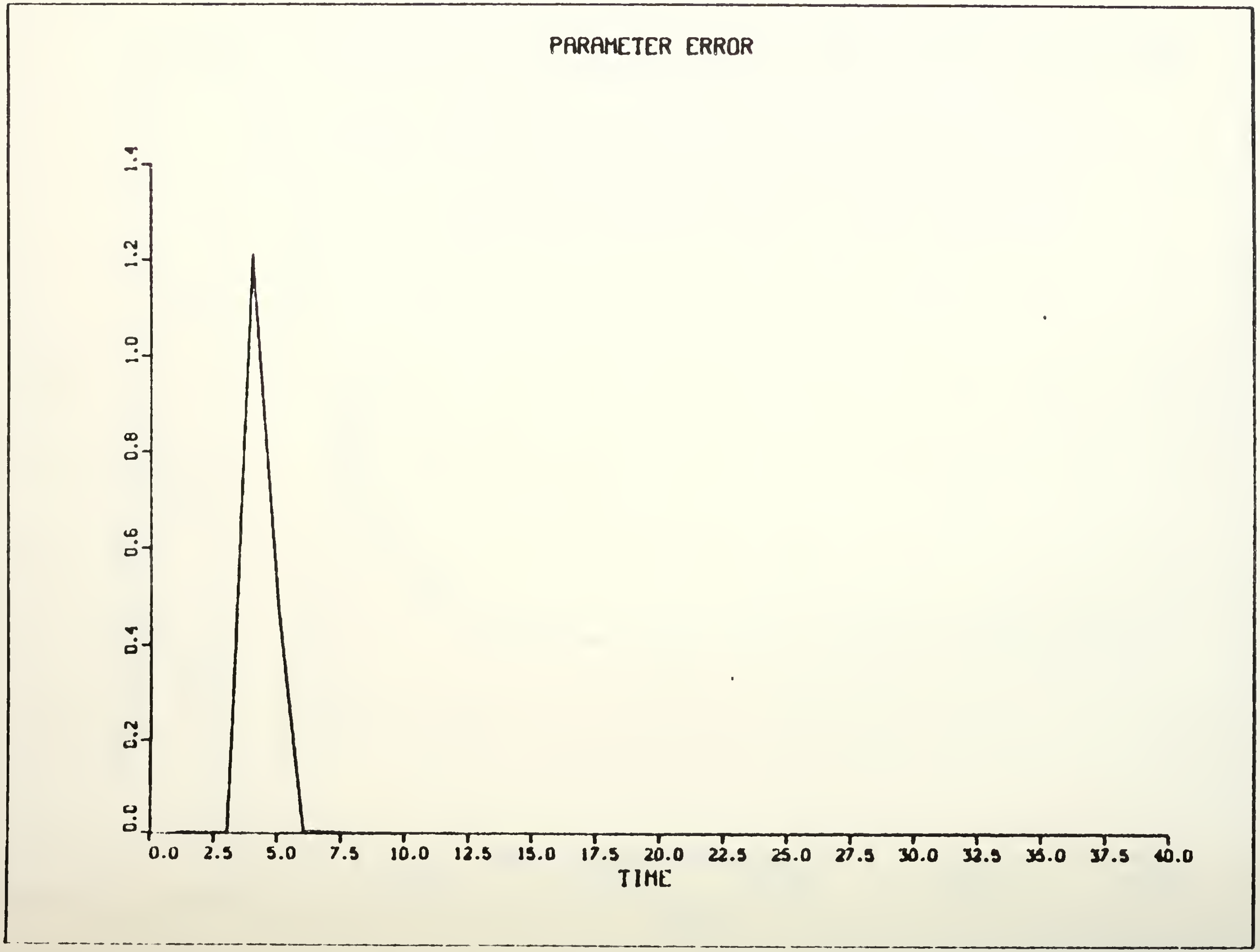

Figure 4.3. Parameter Error vs. Time (Example 1)

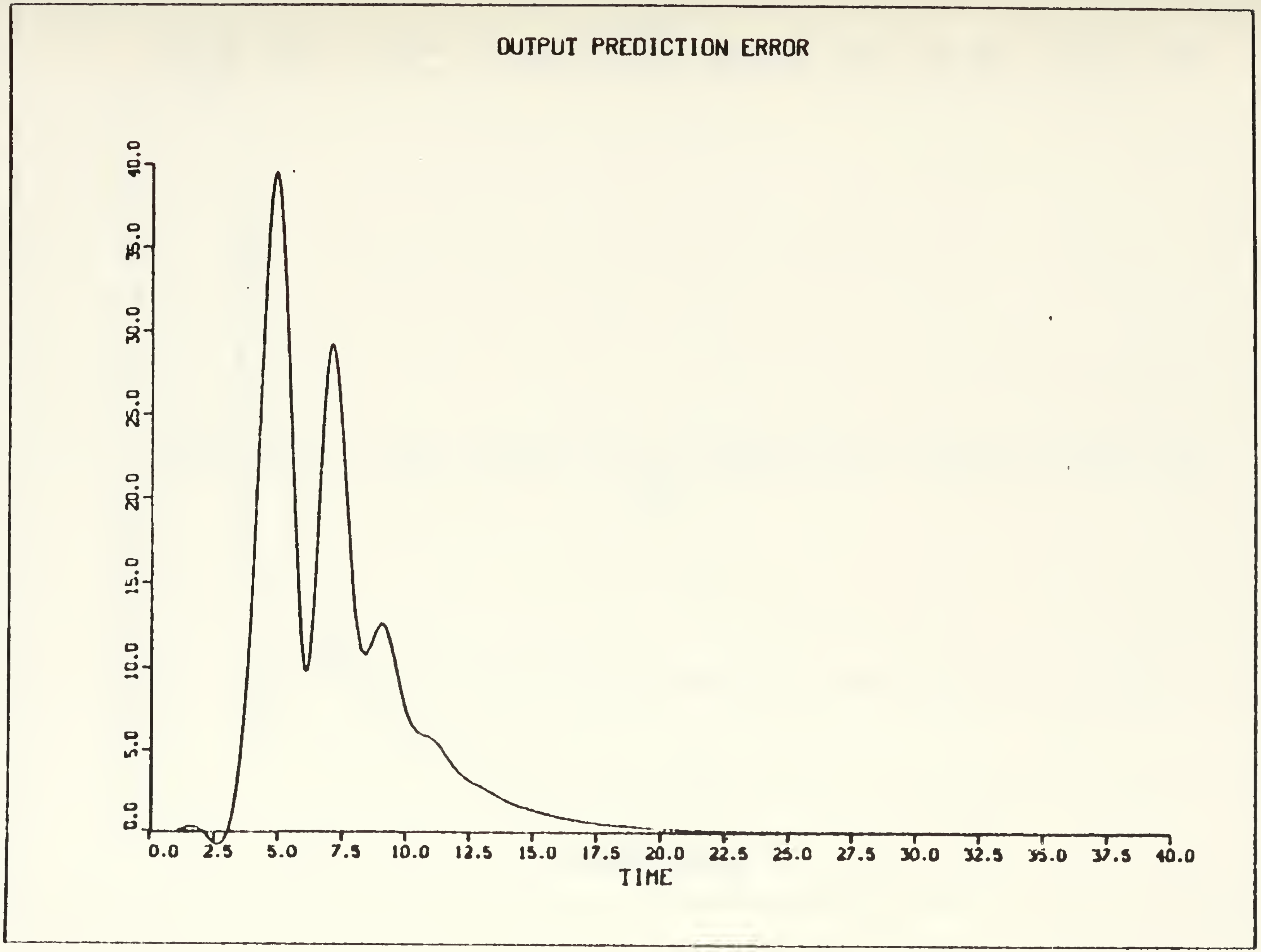

Figure 4.4. Output Prediction Error vs. Time (Example 1)

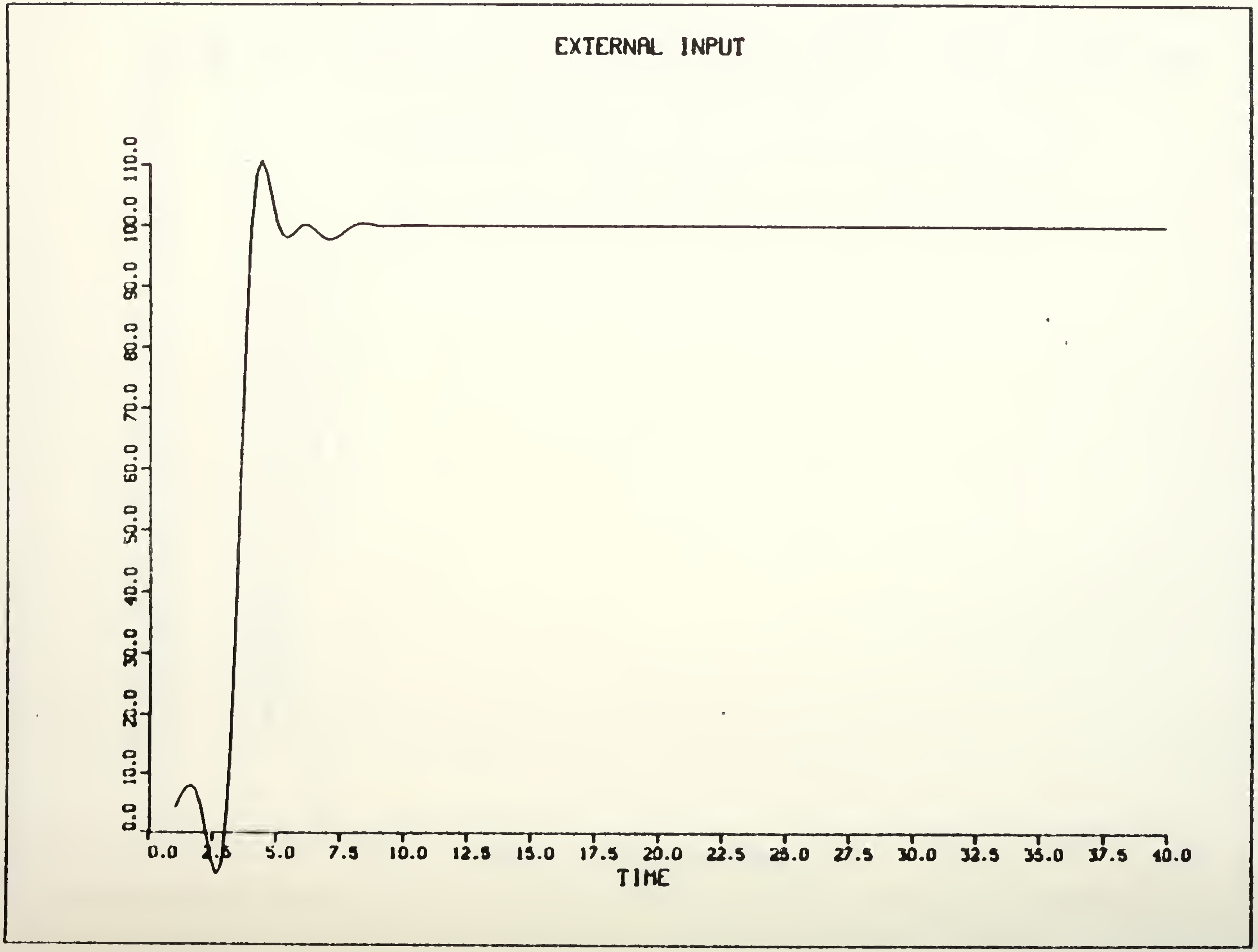

Figure 4.5. External Input vs. Time (Example 1)

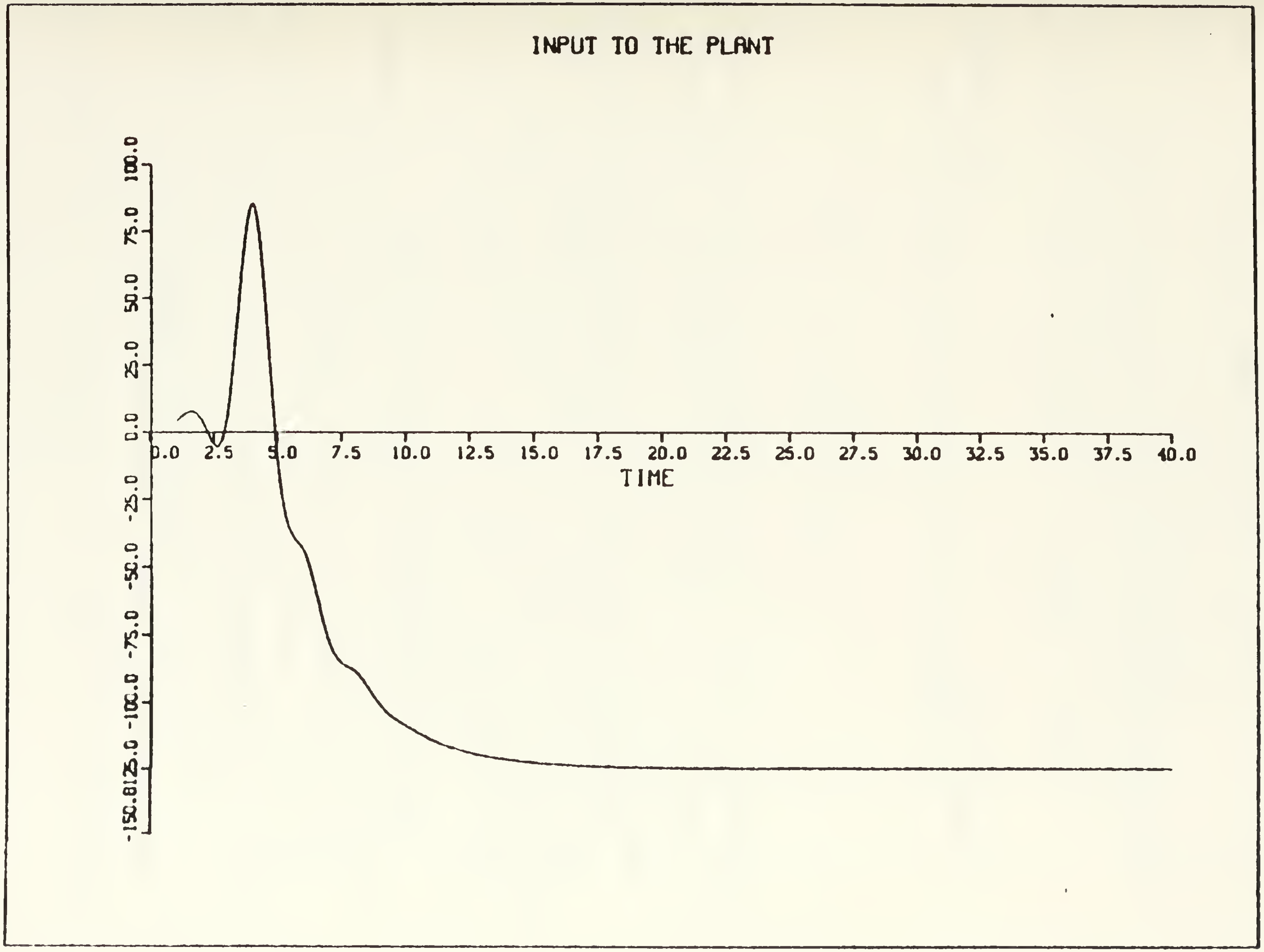

Figure 4.6. Plant's Input vs. Time (Example 1)

perturbed from z = -2 to z = -3. To compute the parameters associated to the perturbed plant with pulse transfer function

$$H(z) = \frac{z + 3}{z^2 - 2z + 0.75}$$

write its difference equation as

$$y(t) = 2y(t-1) - 0.75y(t-1) + u(t-1) + 3u(t-2)$$

and immediately

$$\theta^* = \begin{bmatrix} -2.0 \\ 0.75 \\ 1.0 \\ 3.0 \end{bmatrix}$$

By using the same observer and desired polynomials g(z), p*(z) and following the same steps as in Example 4.1, the controller parameters become $k_1 = -0.894$, $k_o = -0.778$, $h_1 = -0.305$ and $h_o = 0.152$.

Then, the polynomials k(z) and h(z) are:

$$k(z) = -.894z - 0.778$$

$$h(z) = -0.305z + 0.152$$

The closed loop transfer function becomes:

$$H(z) = \frac{z + 3}{z^2 - 1.1z + 0.3}$$

The step response of this transfer function is given in Figure 4.7. Also, the other graphical results are in Figures 4.8-4.12.

In the next example, we investigate how the disturbance at the output will affect the behavior of the controlled system.

Example 4.3:

Consider a plant with pulse transfer function as:

$$H(z) = \frac{z + 2}{z^2 - 2z + 0.75}$$

and assume an output disturbance exists. Let the disturbance be sinusoidal with frequency $5\pi/2$ rad/sec. In this case, it is observed that the estimated parameters of the plant don't converge to the actual parameters. Corresponding plots are given in Figure 4.13 through 4.17. However, if the disturbance is small, the parameters converge close to the actual values and we can still obtain satisfactory performances.

Comparison of Different Estimation Techniques: RLS and P.A.

In the above examples we used a recursive least-squares algorithm for estimating the plant parameters. In this section, we compare the behavior of the indirect adaptive control, when the projection algorithm is used.

The projection algorithm has the advantage of requiring less computations. Approximately, the number of computations

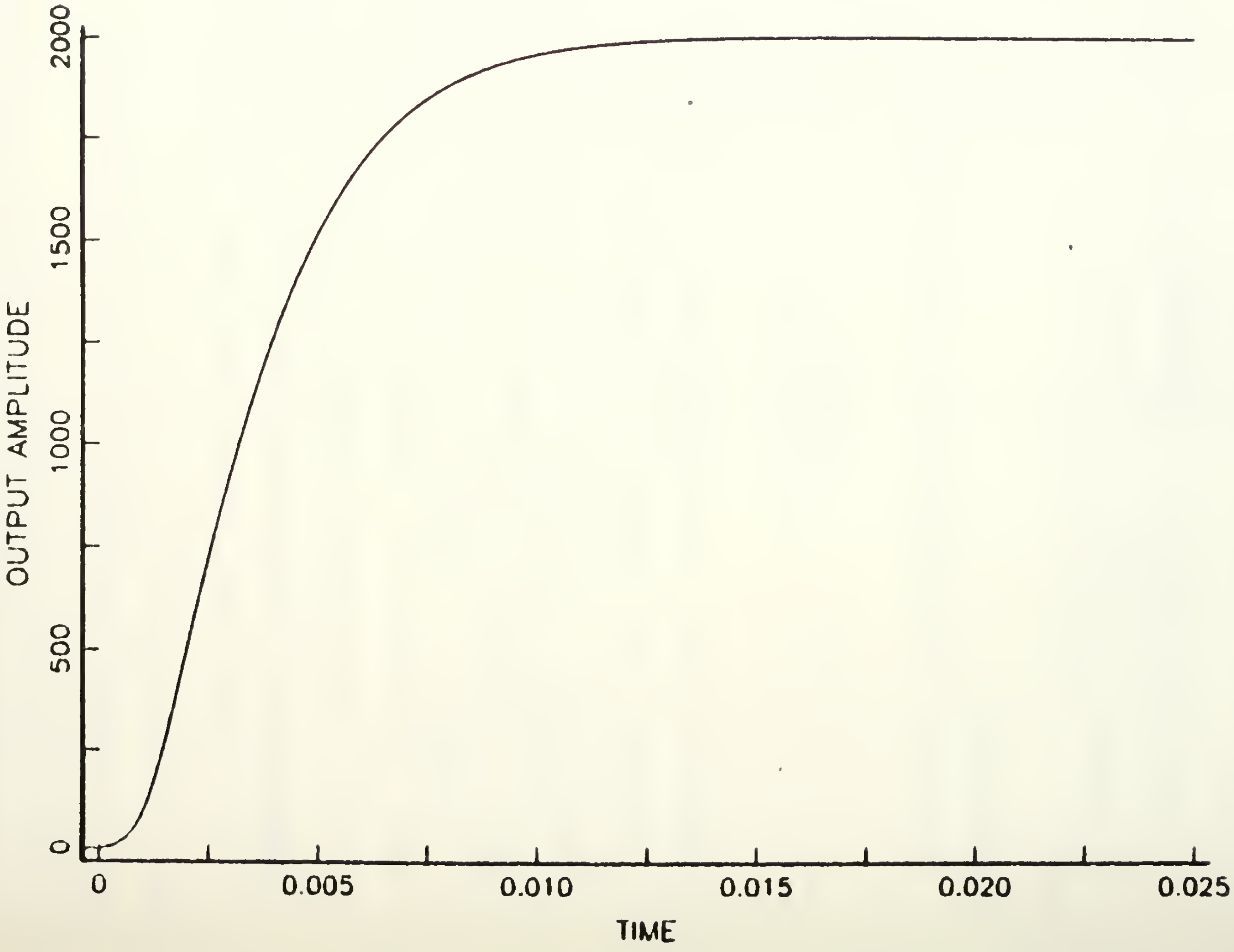

Figure 4.7. Step Response of the Closed-Loop System (Example 2)

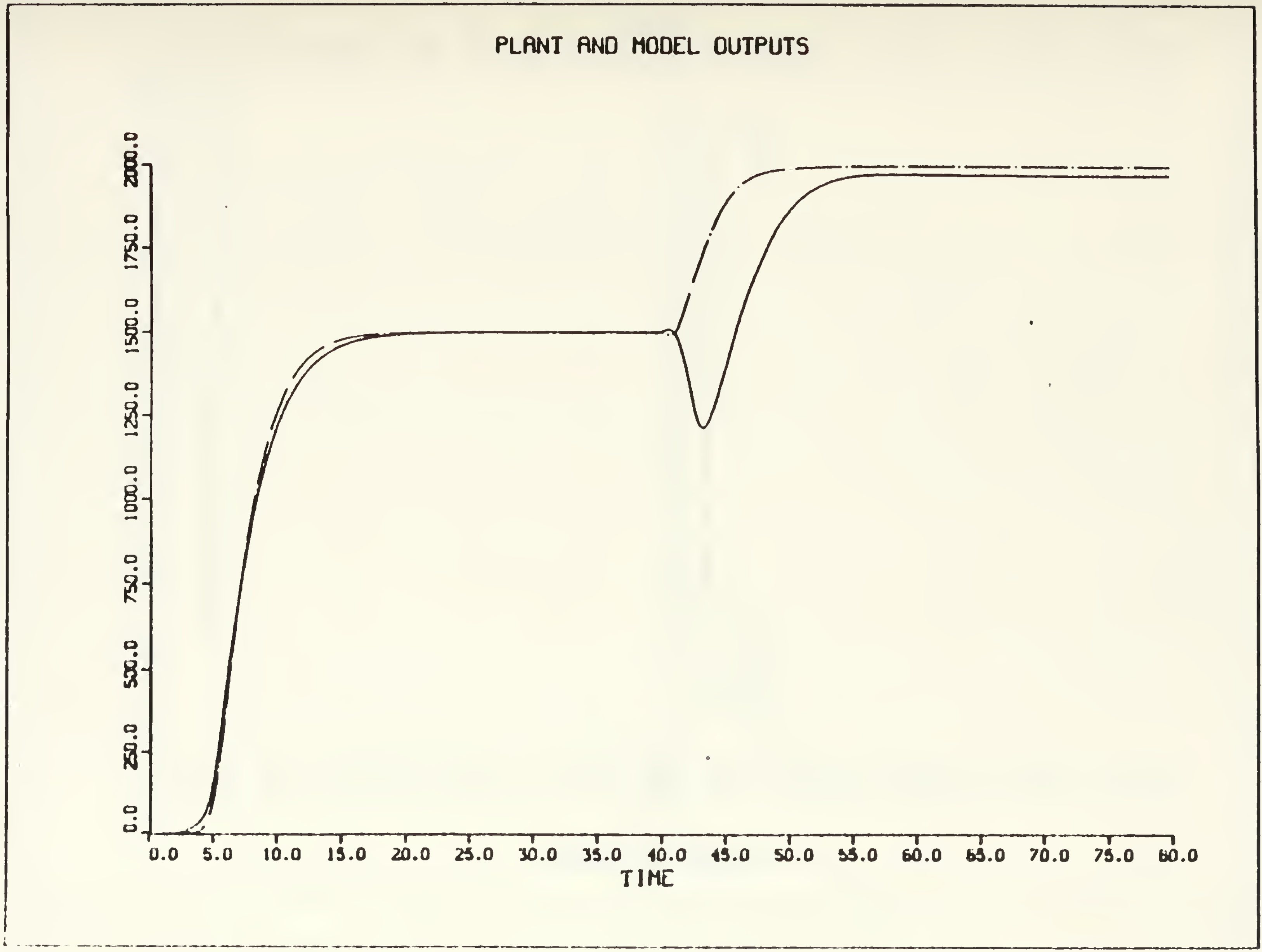

Figure 4.8. Plant and Model Outputs vs. Time (Example 2)

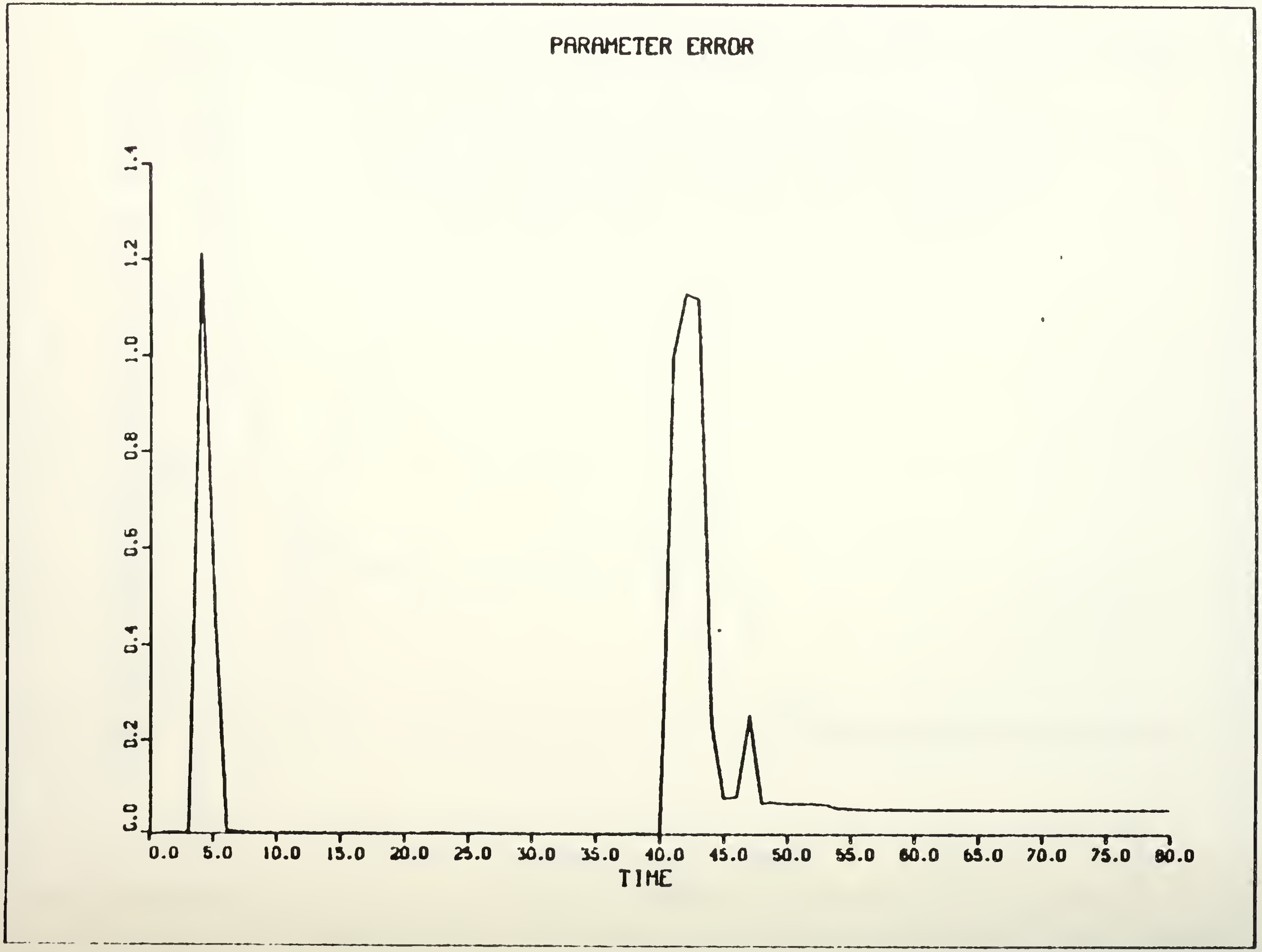

Figure 4.9. Parameter Error vs. Time (Example 2)

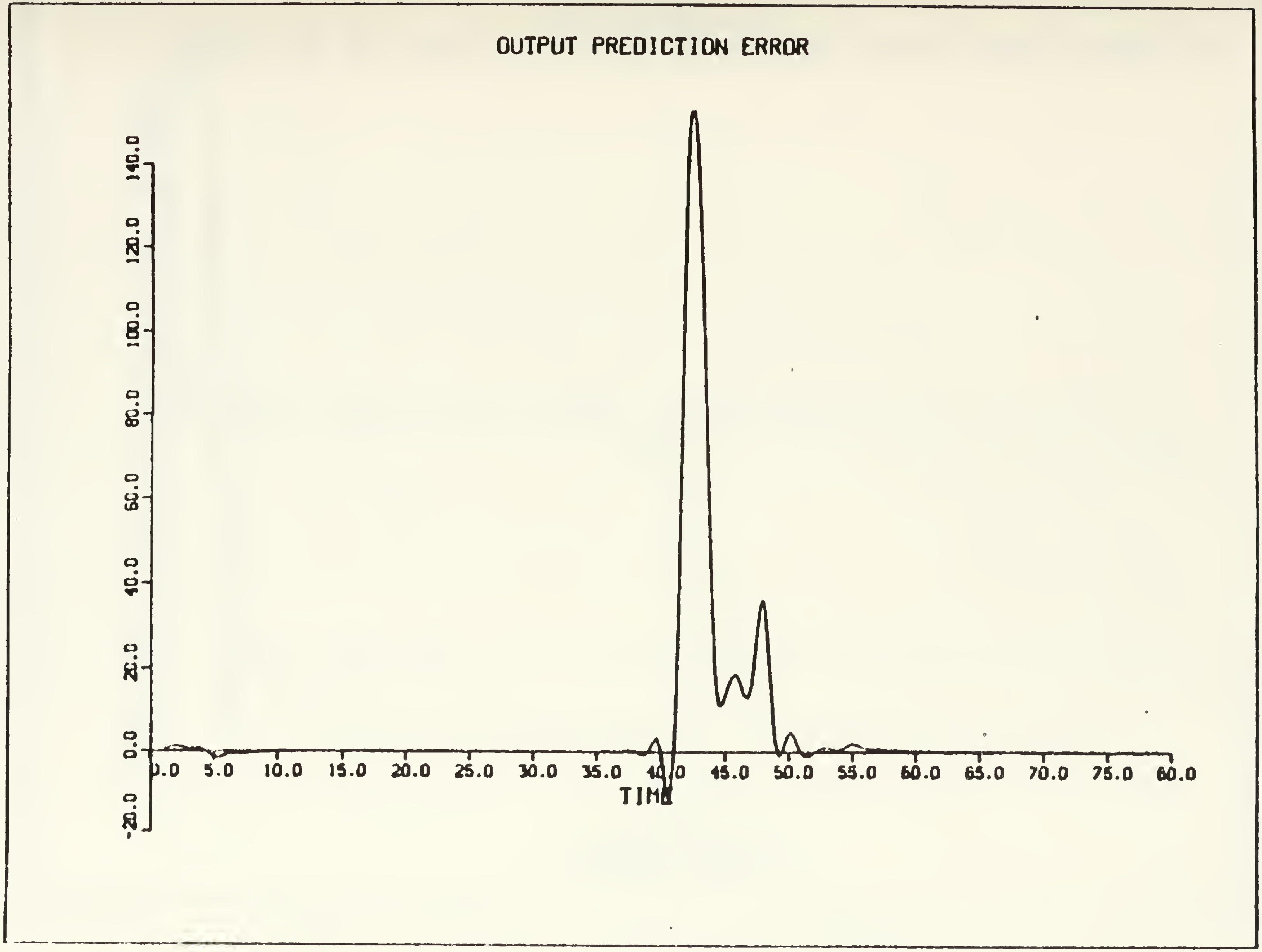

Figure 4.10. Output Prediction Error vs. Time (Example 2)

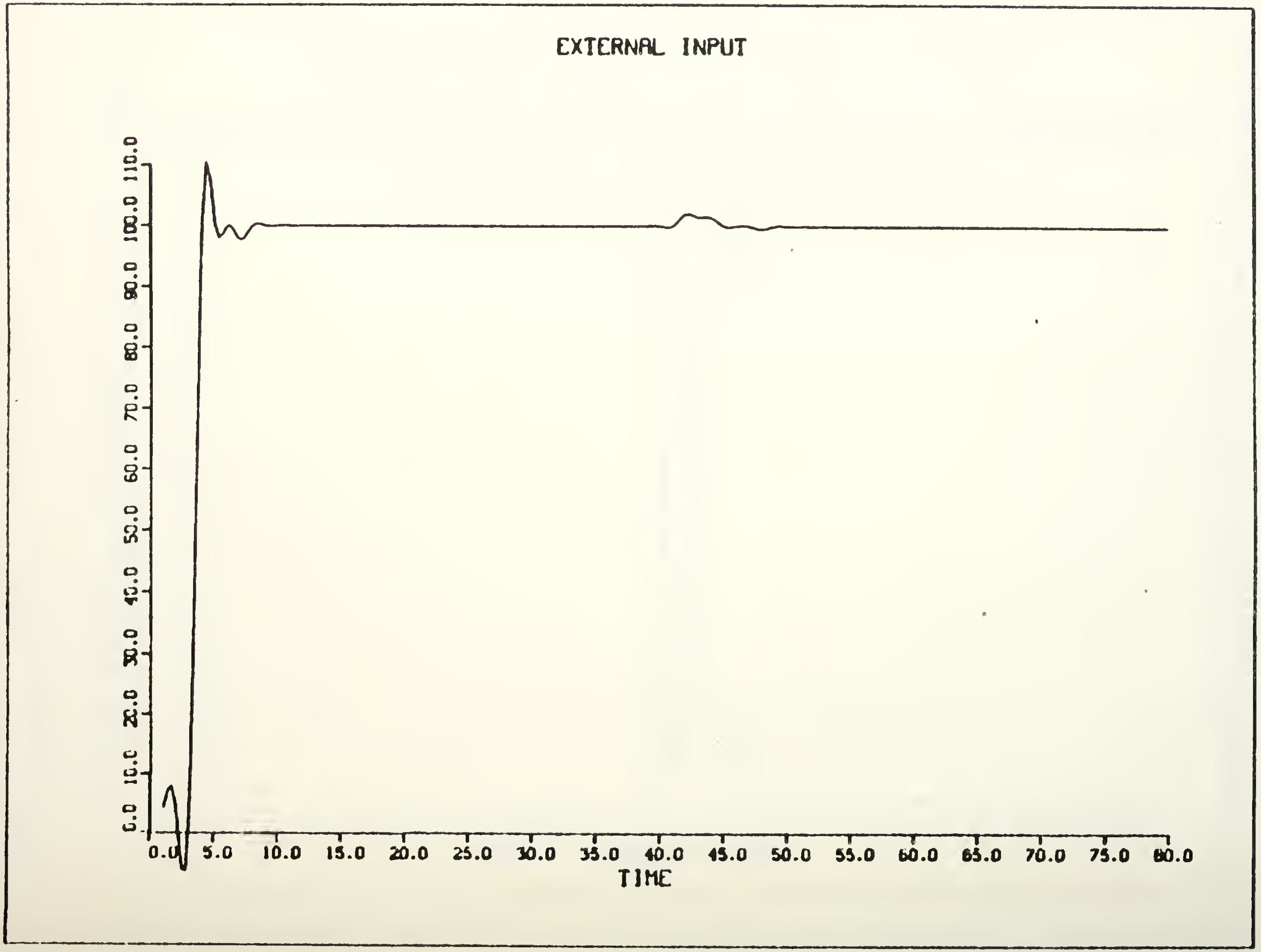

Figure 4.11. External Input vs. Time (Example 2)

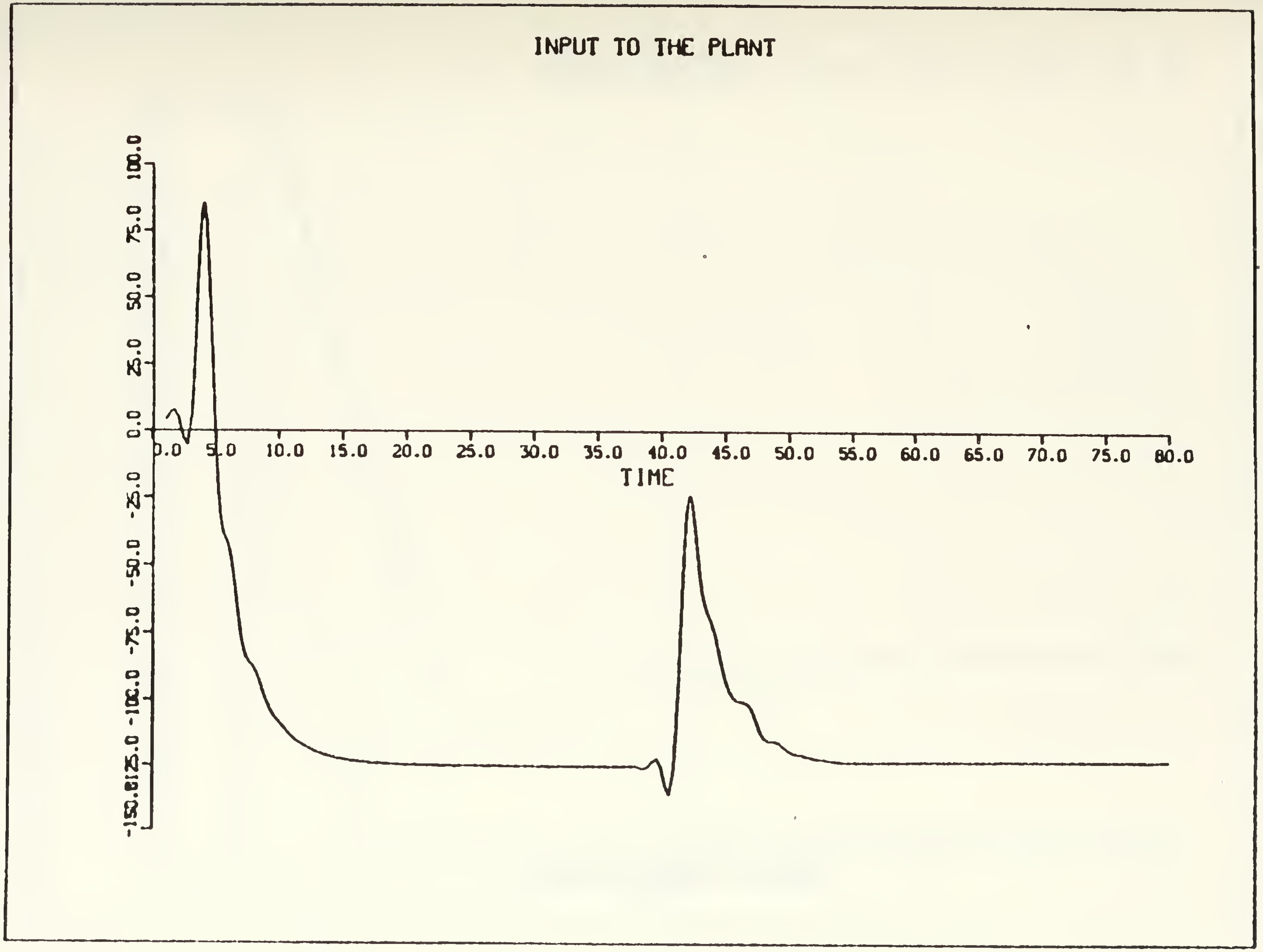
INPUT TO THE PLANT

Figure 4.12. Plant's Input vs. Time (Example 2)

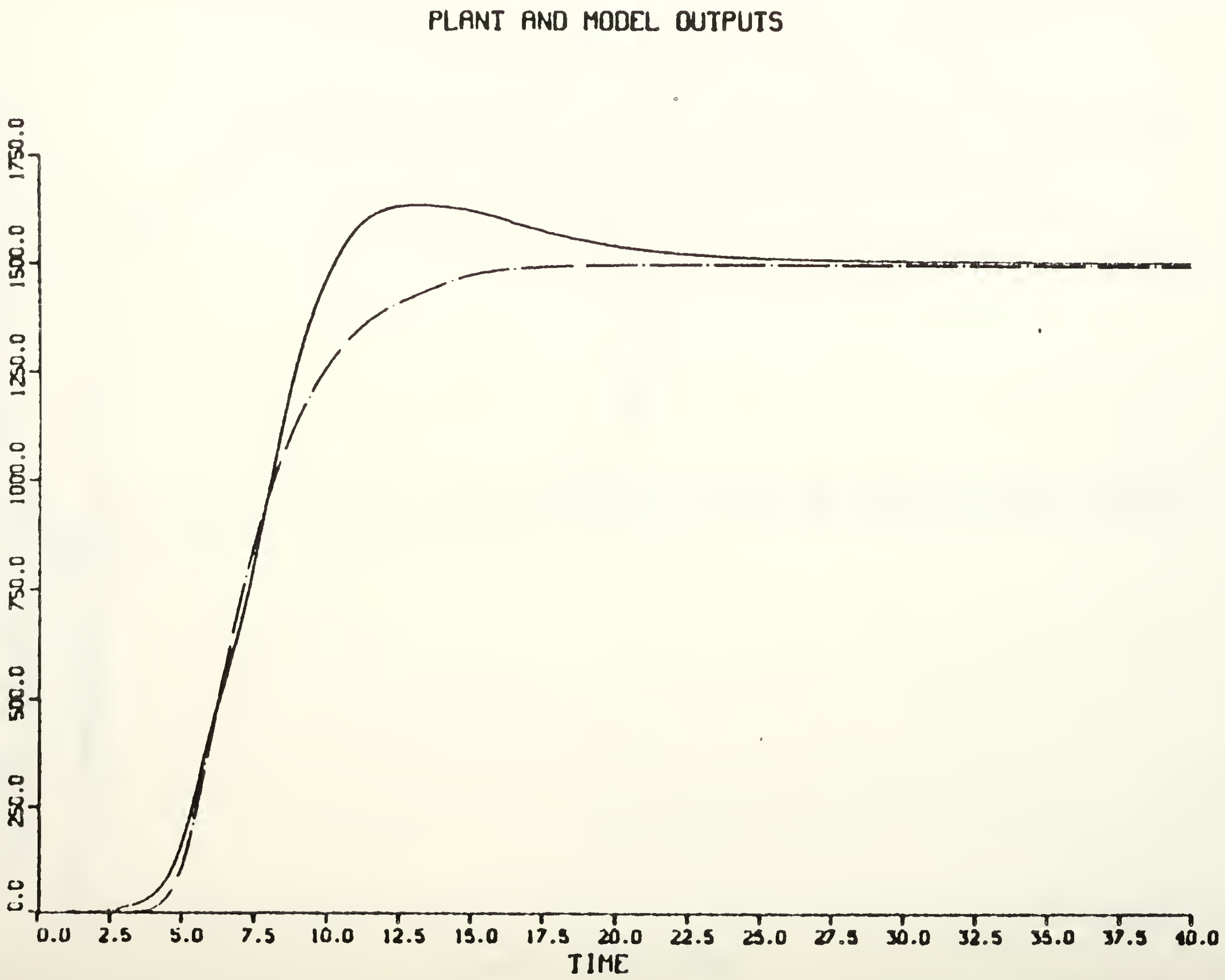

Figure 4.13. Plant and Model Outputs vs. Time (Example 3)

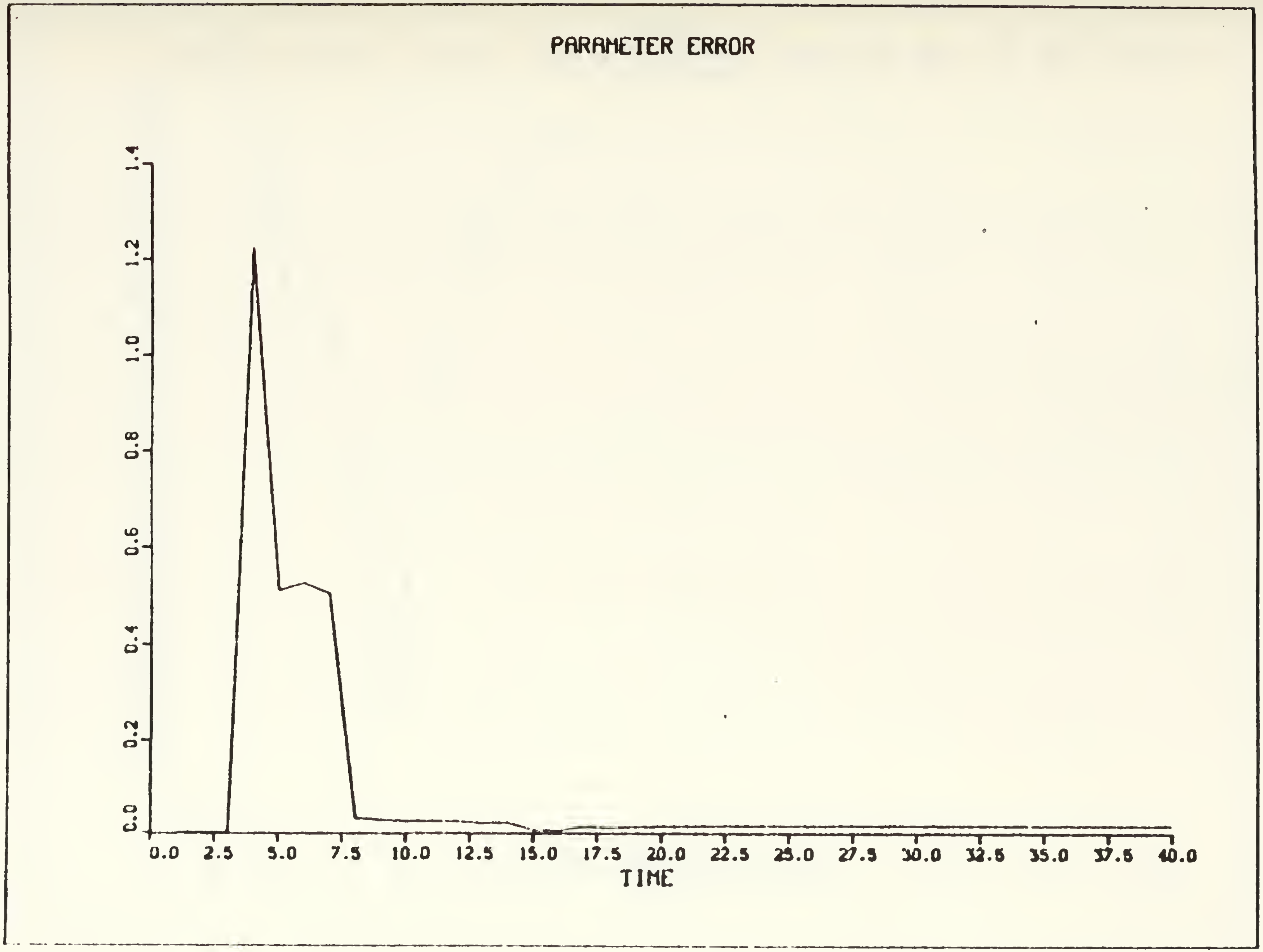

Figure 4.14. Parameter Error vs. Time (Example 3)

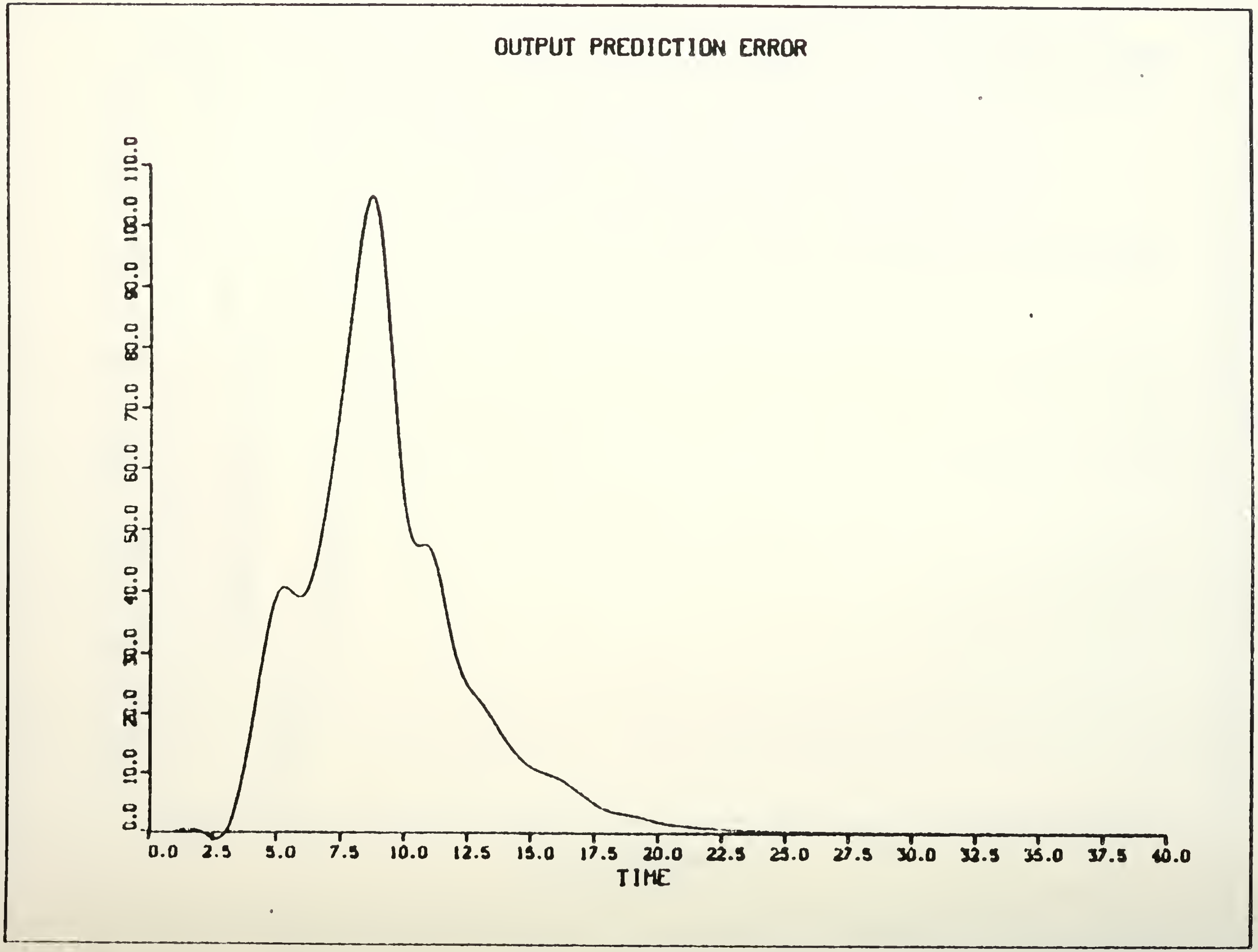

Figure 4.15. Output Prediction Error vs. Time (Example 3)

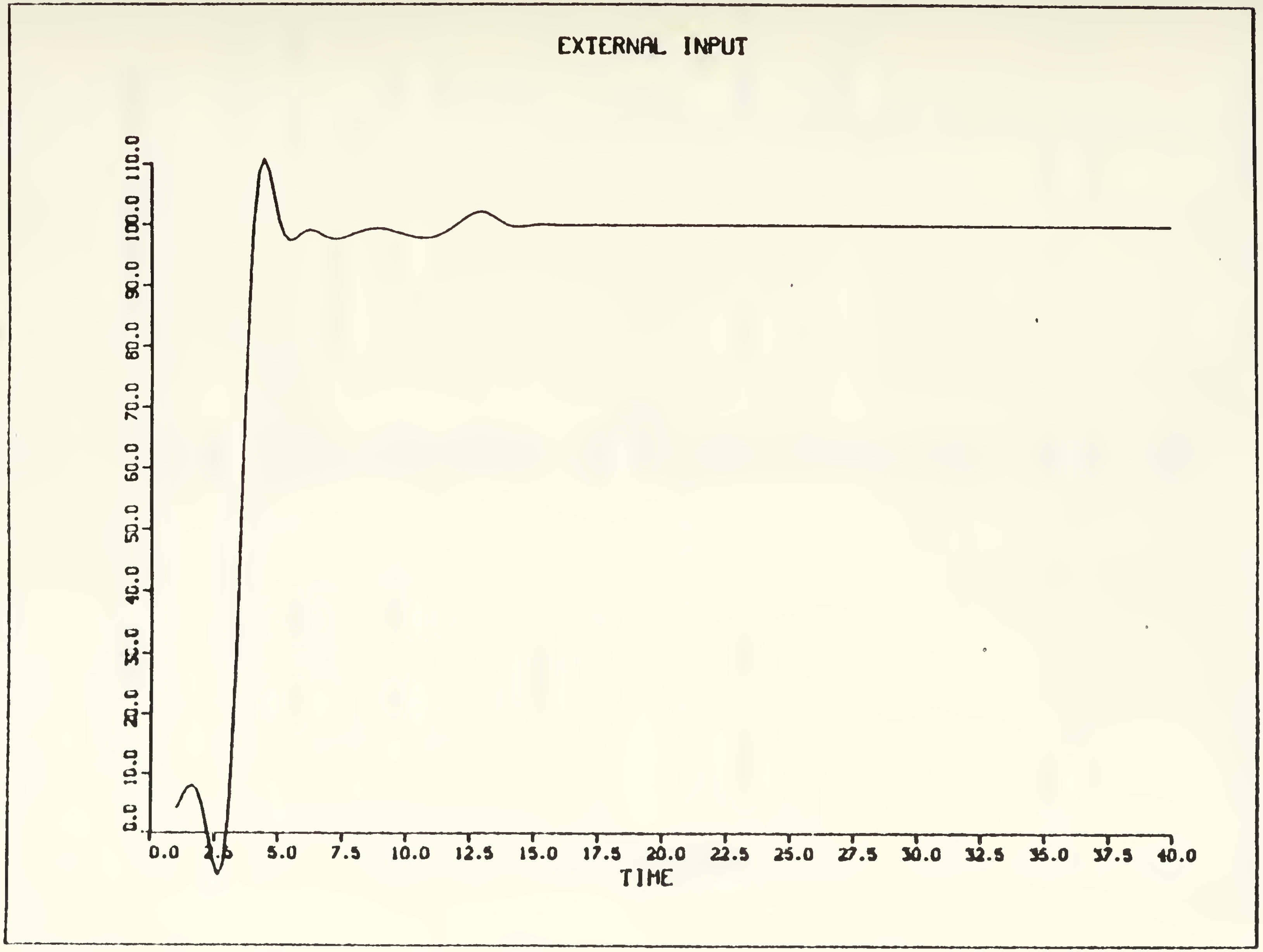

Figure 4.16. External Input vs. Time (Example 3)

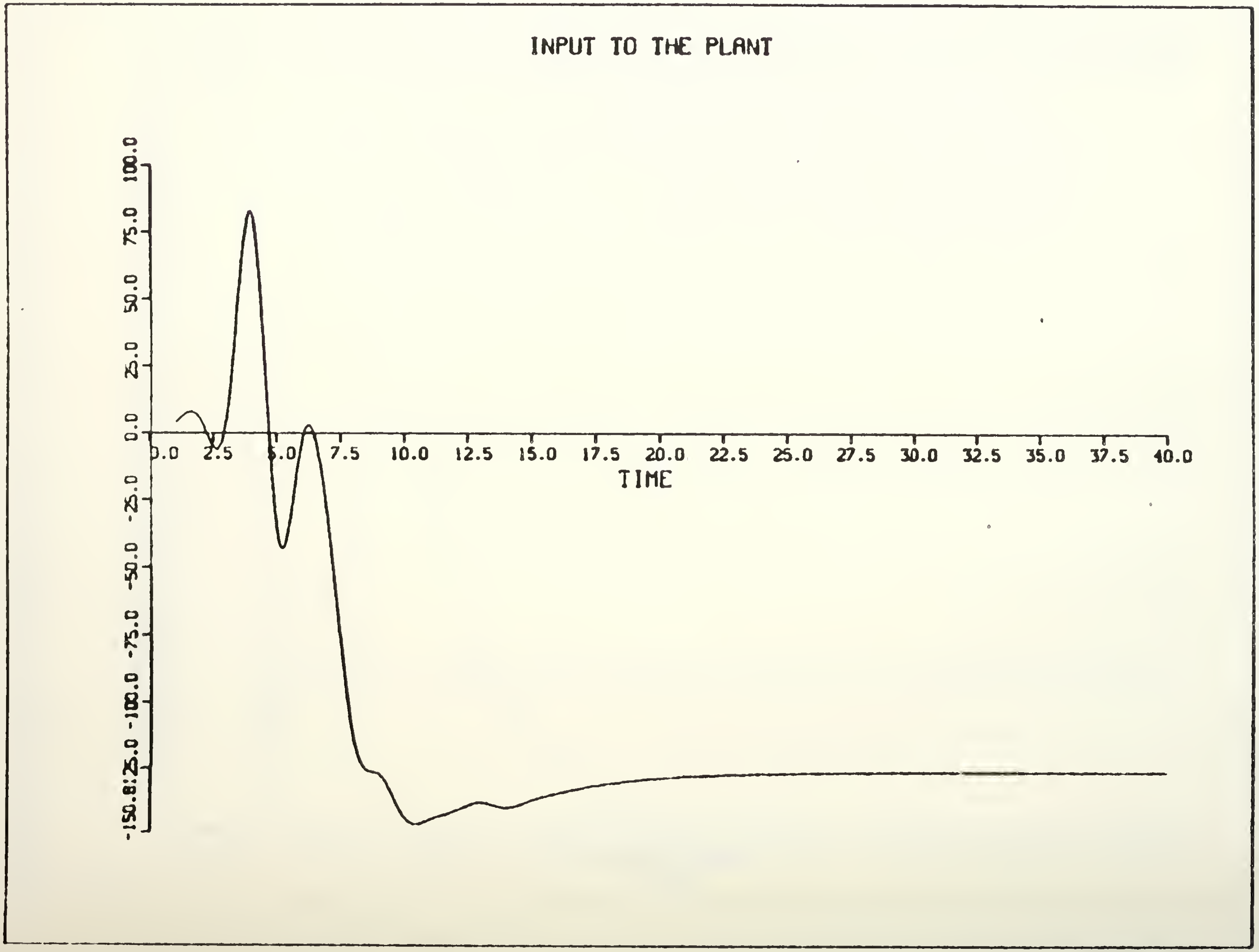

Figure 4.17. Plant's Input vs. Time (Example 3)

grows as n^2 in the recursive least-squares algorithm. But in the projection algorithm, it grows as n. As noted before, the projection algorithm can be described as

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{a\Phi(t-1)}{c + \Phi(t-1)^T\Phi(t-1)}[y(t) - \Phi(t-1)^T\hat{\theta}(t-1)]$$

with $c > 0$ and $0 < a < 2$. The value of the parameter a is crucial. The behavior os the algorithm greatly depends on its value.

In the next example, the projection algorithm is used for estimation procedure on the previous process.

Example 4.4:

Using the same pulse transfer function and controller polynomials as in Example 4.3 and choosing the constants $a = 1$ and $c = 1$ in the above equation, results can be observed from Figure 4.18 through 4.22.

Actually, using this algorithm, several systems have been simulated. This result is the most reasonable one.

Considering the above results, indirect adaptive control is a very effective control technique when the parameters of the plant are unknown and large and unpredictable variations occur in the plant dynamics. Also, it is applicable to nonminimum-phase systems. This can be considered as a big advantage of the method.

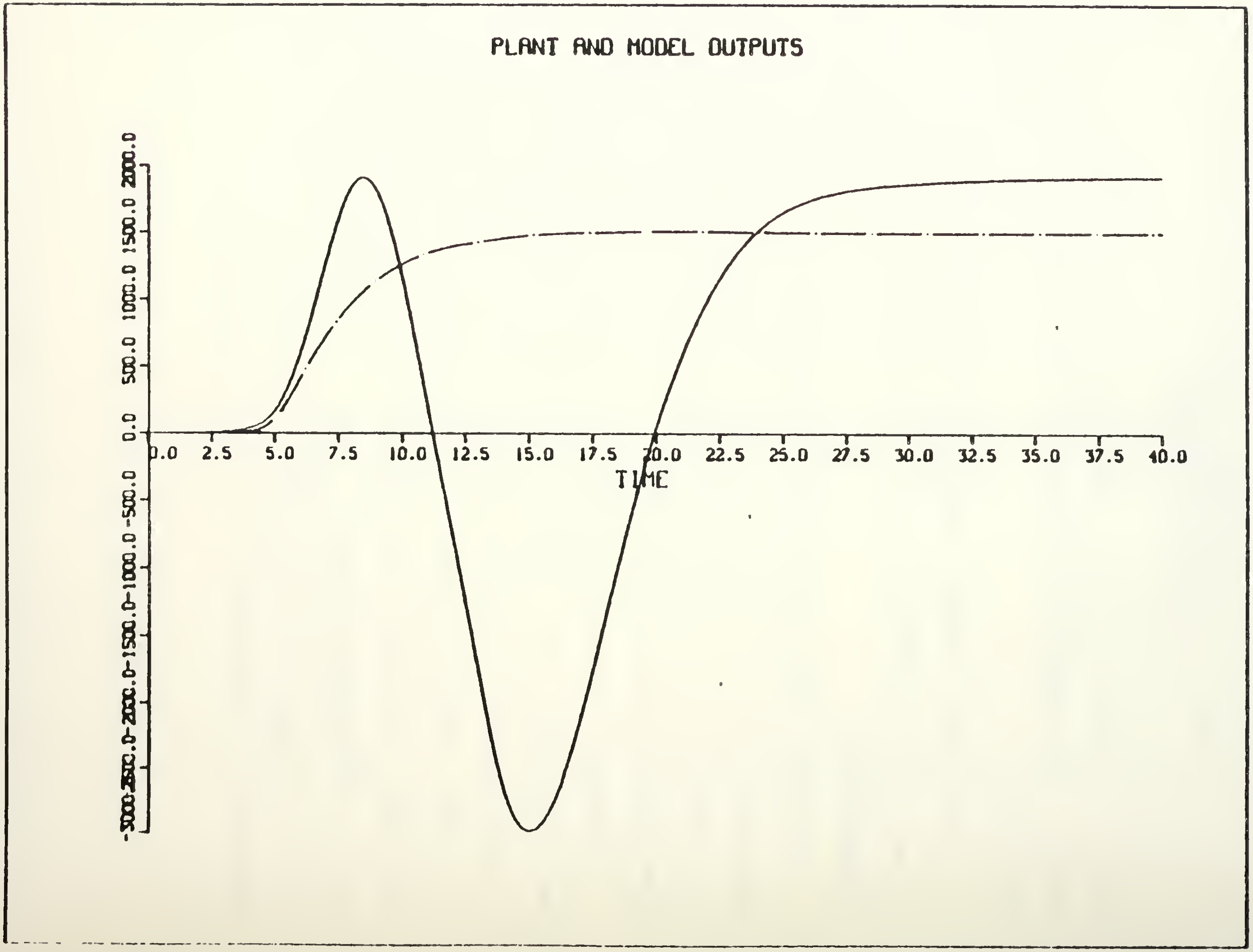

Figure 4.18. Plant and Model Outputs vs. Time (Example 4)

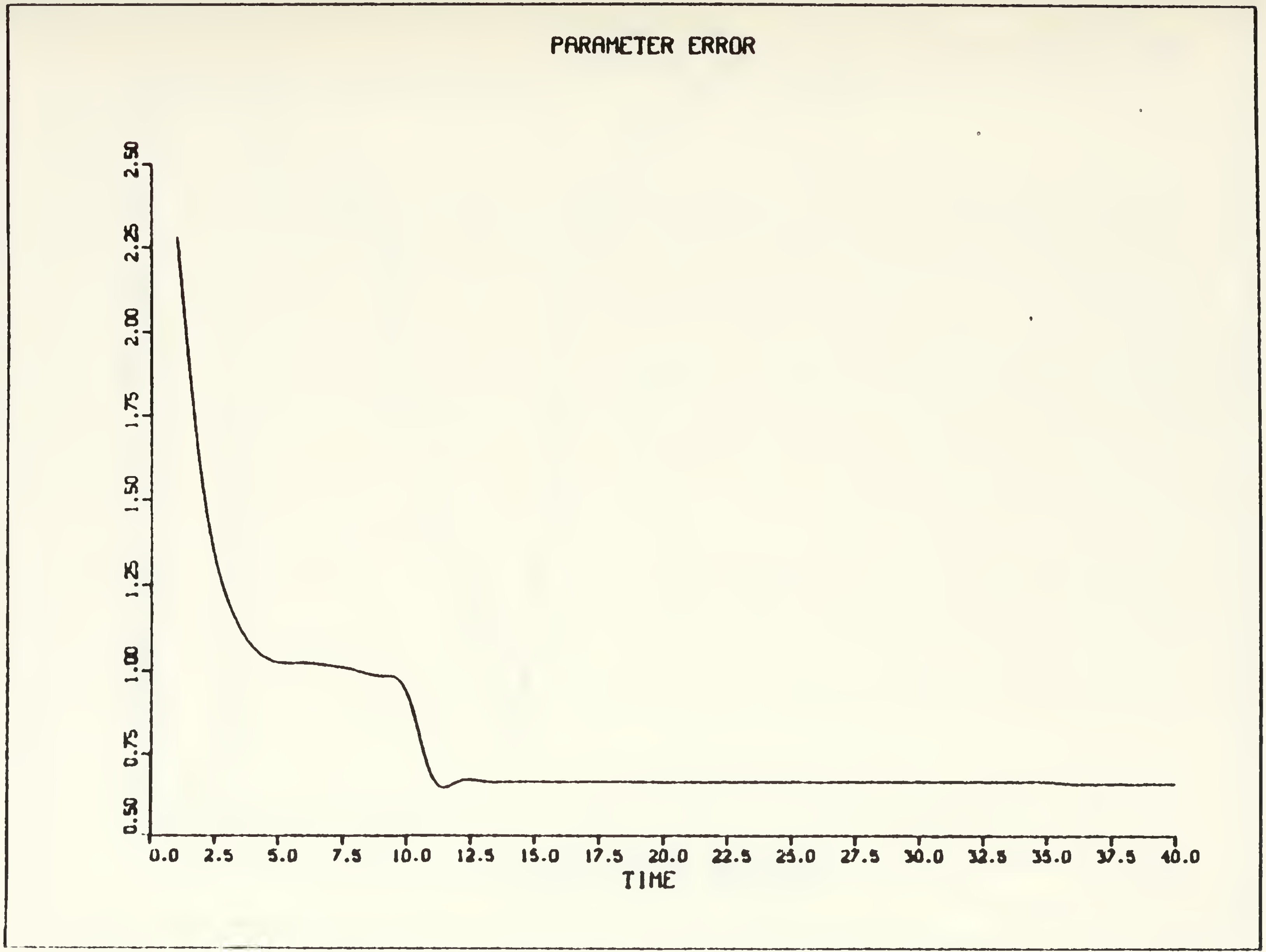

Figure 4.19. Parameter Error vs. Time (Example 4)

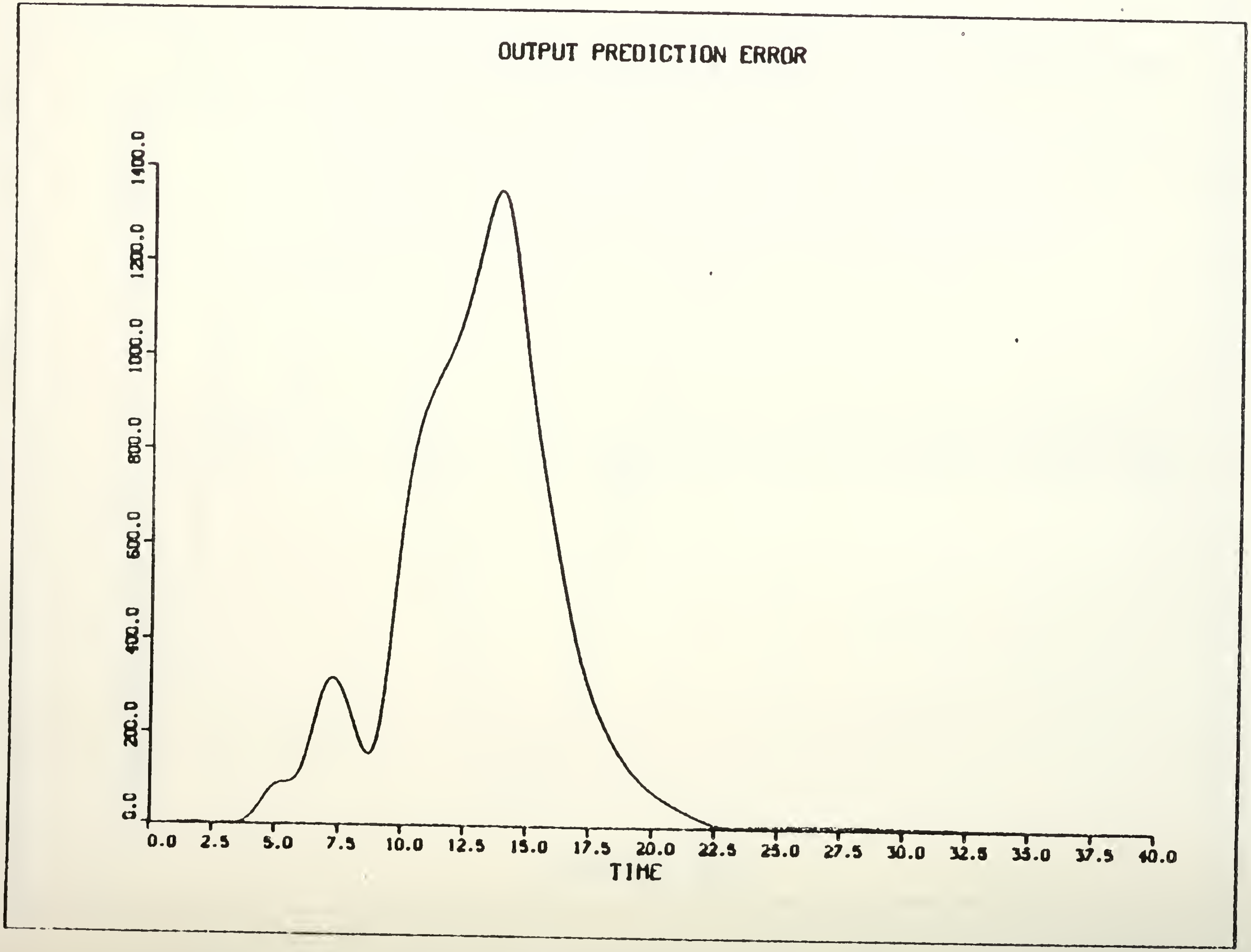

Figure 4.20. Output Prediction Error vs. Time (Example 4)

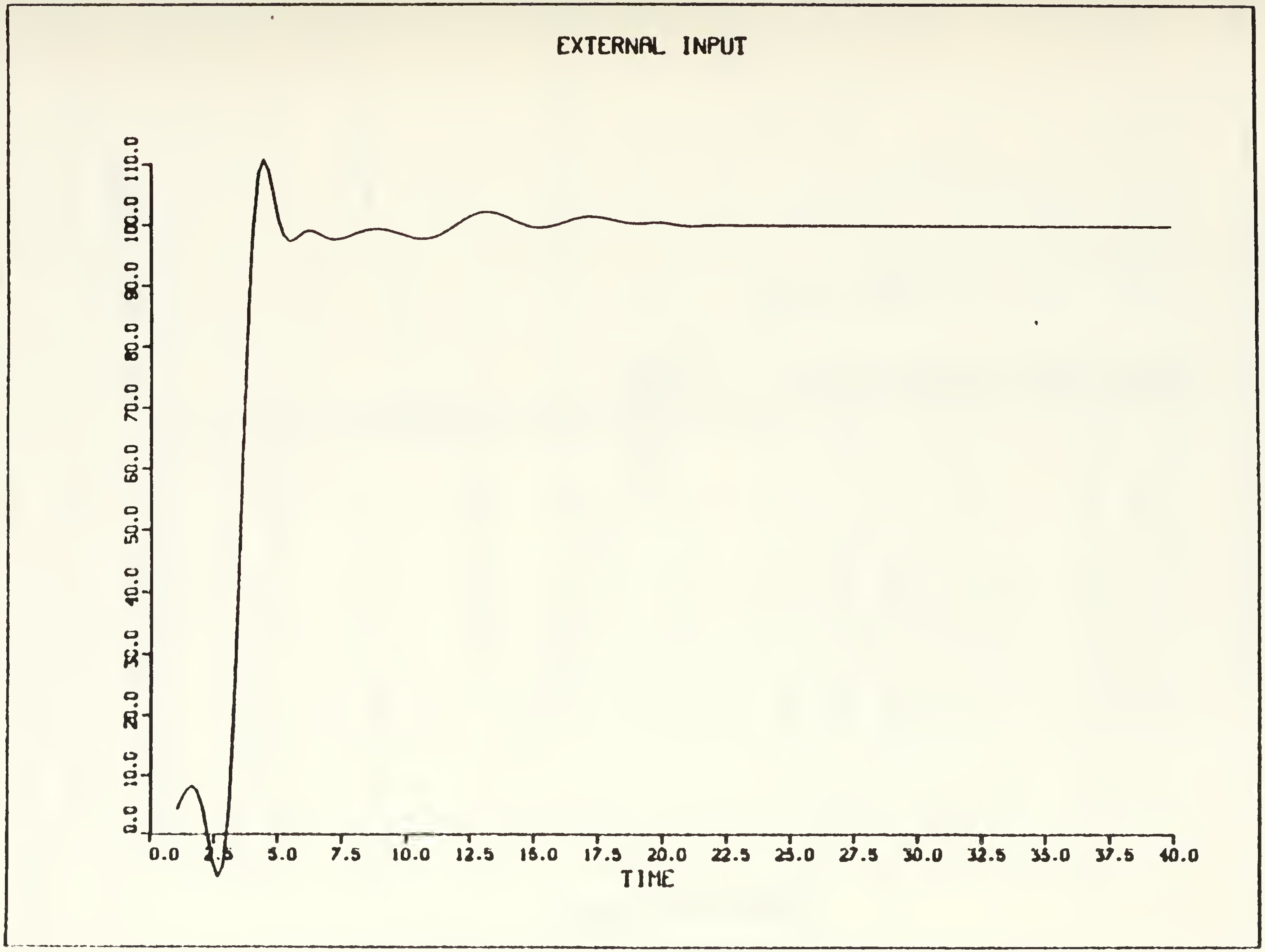

Figure 4.21. External Input vs. Time (Example 4)

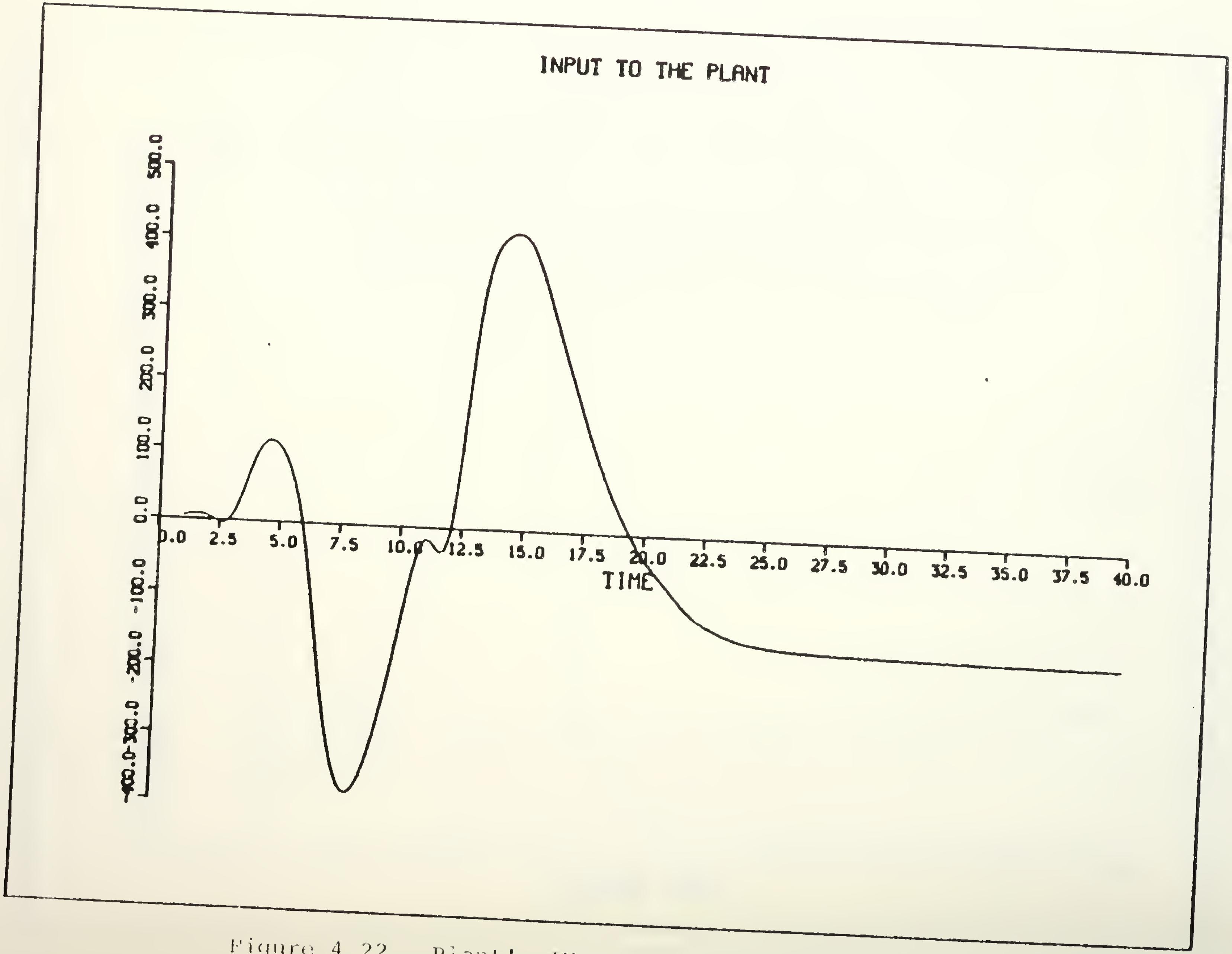

Figure 4.22. Plant's INput vs. Time (Example 4)

APPENDIX A

DESCRIPTION OF COMPUTER PROGRAMS

Computer programs are written using WATFIV and FORTRAN programming languages. All of them are prepared as an interactive program. The following explanation is about how one can use these programs.

The program named CONDIS given in Appendix B finds the pulse transfer function from the continuous time transfer function. The program asks for the sampling interval, orders and coefficients of the continuous time transfer function polynomials. The continuous time system is described in state variable form as

$$\dot{\underline{x}} = A\underline{x} + Bu$$

and the corresponding discrete time system as

$$\underline{x}(k+1) = \Phi\underline{x}(k) + \Gamma u(k)$$

This program gives the Φ and Γ matrices corresponding to A and B.

There is no limitation about the order of the system in all programs. For higher order systems, the dimensions of the declared matrices should be increased.

The program named RCIOP given in Appendix D simulates an indirect adaptive control system using the projection algorithm. The program asks some questions to the user in the following order:

1. The order of a plant (N);
2. The constants a and c that are given in Equation (3.38);
3. Coefficients of the numerator polynomial of the plant (in ascending order of z);
4. Coefficients of the denominator polynial of the plant (in ascending order of z);
5. Coefficients of the controller polynomials g(z) and p*(z).

The external input is defined inside the program. It can be changed when it is necessary.

The last program RCIOR given in Appendix C is the most important one. Least-squares algorithms are used for parameter estimation. This program asks the same questions about system, except question #2.

Computer programs RCIOR and RCIOP give the graphical and tabulated results. Graphical outputs are obtained using DISSPLA.

APPENDIX B

COMPUTER PROGRAM CONDIS

```
$JOB    WATFIV  ONUK,XREF,EXT
C        DEFINITION OF VARIABLES:
        INTEGER N,M,R,I,K,L,L1,L2,I1,J,I3,M1,M2,N3
        integer M3,L3,I6
        REAL AA(2,2),BB(2,1),ID(2,2),PS(10,10)
        REAL FI(10,10),GT(10,10),C1(10,10),C2(10,10)
        real c3(10,10),c8(10,10),d4(10,10),d9(10,10)
        REAL C4(10,10),C5(10,10),C6(10,10),C7(10,10)
        REAL C10(10,10),C11(10,10),D2(10,10),D3(10,10)
        REAL D5(10,10),D6(10,10),D7(10,10),D8(10,10)
        real c14(10,10),c19(10,10),c20(10,10),e4(10,10)
        REAL D10(10,10),D11(10,10),C12(10,10),C13(10,10)
        REAL C15(10,10),C16(10,10),C17(10,10),C18(10,10)
        REAL C110(10,10),W1(1,10)/10*0./,ID1(10,10)
        REAL CC(10),TT(10),DD(10),RR(2,2),PP(2,2)
        REAL D1(10,10),E1(10,10),E2(10,10),E3(10,10)
        REAL E5(10,10),E6(10,10),C9(10,10),ss(2,2)
        REAL FF(2,1),GG(1,1),ALPHA(10),CO(10),ALP,TAR
        REAL TC,TRG,TR1,TR2,TR3,TR4,TR5,TR6,TR7,TR8
        real TR9,TR10,TR11,TK1,c21(10,10)
        PRINT,'THIS PROGRAM FINDS THE Z-DOMAIN PULSE TRANSFER'
        PRINT,'FUNCTION WHEN GIVEN THE S-DOMAIN TRANSFER FUNC.'
        PRINT,' '
        PRINT,'THE S-DOMAIN TRANSFER FUNCTION SHOULD BE
              IN THIS FORM'
C
C       H(S)=NUM(S)/DEN(S)
C           NUM(S)=T(M)S**N-1+.........+T(2)S+T(1)
C           DEN(S)=S**N+D(N)S**N-1+....+D(2)S+D(1)
        PRINT,' '
        PRINT,'ENTER THE NUMERATOR POLYNOMIAL DEGREE'
        READ,M1
        PRINT,' '
        PRINT,'ENTER THE DENOMINATOR POLYNOMIAL DEGREE'
        READ,N
        M2=M1+1
        DO 2 I1=1,M2
           PRINT,'T(',I1,')=?'
           READ,TT(I1)
2       CONTINUE
        N3=N-1
        M3=M1+2
        IF (M1.LT.N3) THEN
```

```
         DO 82 I1=M3,N
            TT(I1)=0.0
82       CONTINUE
      END IF
      DO 12 I1=1,N
         PRINT,'D(',I1,')=?'
         READ,DD(I1)
12    CONTINUE
      DO 22 I1=1,N3
        J=1
        AA(I1,J)=0.0
22    CONTINUE
      DO 32 J=2,N
         DO 102 I1=1,N3
            I3=I1+1
            IF (J.EQ.I3) THEN
              AA(I1,J)=1.0
            ELSE
              AA(I1,J)=0.0
            END IF
102      CONTINUE
32    CONTINUE
      DO 72 J=1,N
        AA(N,J)=-DD(J)
72    CONTINUE
      DO 52 I1=1,N3
        J=1
        BB(I1,J)=0.0
52    CONTINUE
      BB(N,J)=1.0
      DO 62 I1=1,N
         CC(I1)=TT(I1)
62    CONTINUE
      R=1
      M=N
      CALL FAMILY(N,M,'AA')
      CALL RESULT(N,M,I,K,AA)
      CALL FAMILY(N,R,'BB')
      CALL RESULT(N,R,I1,L3,BB)
      CALL FAMILY(R,N,'CC')
      CALL RESULT(R,N,I,K,CC)
      DO 14 J=1,N
         DO 15 I=1,N
            IF (I.EQ.J) THEN
               ID(I,J)=1.0
            ELSE
               ID(I,J)=0.0
            END IF
15       CONTINUE
14    CONTINUE
```

```
       CALL FAMILY(N,M,'ID')
       CALL RESULT(N,M,I,J,ID)
68       M=1
         DO 285 I=1,N
            DO 286 J=1,N
               RR(I,J)=ID(I,J)
286         CONTINUE
285      CONTINUE
          TR1=1.0
          CALL ALI(N,N,I,K,AA,W1)
          GAMMA=AMAX1(W1(1,1),W1(1,2),W1(1,3),W1(1,4)
     *    w1(1,5),w1(1,6),W1(1,7),W1(1,8),W1(1,9),W1(1,10))
          TC=1/GAMMA
         PRINT,TC
         TH1=TR1/2
         PRINT,TH1
         TH2=TR1/4
         PRINT,TH2
         TH3=TR1/8
         PRINT,TH3
         TH4=TR1/16
         PRINT,TH4
         IF (TR1.LE.TC)  THEN
         TR2=TR1/2
         TR3=TR1/3
         TR4=TR1/4
         TR5=TR1/5
         TR6=TR1/6
         TR7=TR1/7
         TR8=TR1/8
         TR9=TR1/9
         TR10=TR1/10
         TR11=TR1/11
         CALL SCALAR(N,N,I,K,TR1,ID,C1)
         CALL SCALAR(N,N,I,K,TR2,AA,D2)
         CALL CALCUL(N,N,N,I,K,K,C2,C1,D2)
         CALL SCALAR(N,N,I,K,TR3,AA,D3)
         CALL CALCUL(N,N,N,I,K,K,C3,C2,D3)
         CALL SCALAR(N,N,I,K,TR4,AA,D4)
         CALL CALCUL(N,N,N,I,K,K,C4,C3,D4)
         CALL SCALAR(N,N,I,K,TR5,AA,D5)
         CALL CALCUL(N,N,N,I,K,K,C5,C4,D5)
         CALL SCALAR(N,N,I,K,TR6,AA,D6)
         CALL CALCUL(N,N,N,I,K,K,C6,C5,D6)
         CALL SCALAR(N,N,I,K,TR7,AA,D7)
         CALL CALCUL(N,N,N,I,K,K,C7,C6,D7)
         CALL SCALAR(N,N,I,K,TR8,AA,D8)
         CALL CALCUL(N,N,N,I,K,K,C8,C7,D8)
         CALL SCALAR(N,N,I,K,TR9,AA,D9)
         CALL CALCUL(N,N,N,I,K,K,C9,C8,D9)
```

```
        CALL SCALAR(N,N,I,K,TR10,AA,D10)
        CALL CALCUL(N,N,N,I,K,K,C10,C9,D10)
        CALL SCALAR(N,N,I,K,TR11,AA,D11)
        CALL CALCUL(N,N,N,I,K,K,C11,C10,D11)
        CALL SUM(N,N,I,K,C1,C2,C12)
        CALL SUM(N,N,I,K,C12,C3,C13)
        CALL SUM(N,N,I,K,C13,C4,C14)
        CALL SUM(N,N,I,K,C14,C5,C15)
        CALL SUM(N,N,I,K,C15,C6,C16)
        CALL SUM(N,N,I,K,C16,C7,C17)
        CALL SUM(N,N,I,K,C17,C8,C18)
        CALL SUM(N,N,I,K,C18,C9,C19)
        CALL SUM(N,N,I,K,C19,C10,C110)
        CALL SUM(N,N,I,K,C110,C11,PS)
        CALL CALCUL(N,N,N,I,K,K,C21,AA,PS)
        CALL SUM(N,N,I,K,ID,C21,FI)
        CALL FAMILY(N,N,'PS')
        CALL RESULT(N,N,I,K,PS)
        CALL FAMILY(N,N,'FI')
        CALL RESULT(N,N,I,K,FI)
        CALL CALCUL(N,R,N,I,K,K,GT,PS,BB)
        CALL FAMILY(N,R,'GT')
        CALL RESULT(N,R,I,L,GT)
        ELSE
        TRG=TR1
        L1=0.0
        TK2=2.0
677     TRG=TR1/(2**L1)
        IF (TC.LE.TRG) THEN
        L1=L1+1
        GO TO 677
        END IF
        PRINT,L1
        TR=TR1/(2**L1)
        CALL SCALAR(N,N,I,K,TR,ID,C1)
        TR2=(TR**2)/2
        CALL SCALAR(N,N,I,K,TR2,AA,C2)
        TR3=(TR**3)/6
        CALL CALCUL(N,N,N,I,K,K,D1,AA,AA)
        CALL SCALAR(N,N,I,K,TR3,D1,C3)
        TR4=(TR**4)/24
        CALL CALCUL(N,N,N,I,K,K,D2,AA,D1)
        CALL SCALAR(N,N,I,K,TR4,D2,C4)
        TR5=(TR**5)/120
        CALL CALCUL(N,N,N,I,K,K,D3,AA,D2)
        CALL SCALAR(N,N,I,K,TR5,D3,C5)
        TR6=(TR**6)/720
        CALL CALCUL(N,N,N,I,K,K,D4,AA,D3)
        CALL SCALAR(N,N,I,K,TR6,D4,C6)
        TR7=(TR**7)/5040
```

```
         CALL CALCUL(N,N,N,I,K,K,D5,AA,D4)
         CALL SCALAR(N,N,I,K,TR7,D5,C7)
         TR8=(TR**8)/40320
         CALL CALCUL(N,N,N,I,K,K,D6,AA,D5)
         CALL SCALAR(N,N,I,K,TR8,D6,C8)
         TR9=(TR**9)/362880
         CALL CALCUL(N,N,N,I,K,K,D7,AA,D6)
         CALL SCALAR(N,N,I,K,TR9,D7,C9)
         TR10=(TR**10)/3628800
         CALL CALCUL(N,N,N,I,K,K,D8,AA,D7)
         CALL SCALAR(N,N,I,K,TR10,D8,C10)
         TR11=(TR**11)/39916800
         CALL CALCUL(N,N,N,I,K,K,D9,AA,D8)
         CALL SCALAR(N,N,I,K,TR11,D9,C11)
         CALL SUM(N,N,I,K,C1,C2,C12)
         CALL SUM(N,N,I,K,C12,C3,C13)
         CALL SUM(N,N,I,K,C13,C4,C14)
         CALL SUM(N,N,I,K,C14,C5,C15)
         CALL SUM(N,N,I,K,C15,C6,C16)
         CALL SUM(N,N,I,K,C16,C7,C17)
         CALL SUM(N,N,I,K,C17,C8,C18)
         CALL SUM(N,N,I,K,C18,C9,C19)
         CALL SUM(N,N,I,K,C19,C10,C20)
         CALL SUM(N,N,I,K,C20,C11,PS)
         CALL FAMILY(N,N,'P1')
         CALL RESULT(N,N,I,K,PS)
         DO 255 I6=1,L1
            CALL SCALAR(N,N,I,K,TK2,ID,E1)
            CALL FAMILY(N,N,'E1')
            CALL RESULT(N,N,I,K,E1)
            CALL CALCUL(N,N,N,I,K,K,E2,AA,PS)
            CALL SUM(N,N,I,K,E1,E2,E3)
            CALL FAMILY(N,N,'E3')
            CALL RESULT(N,N,I,K,E3)
            CALL CALCUL(N,N,N,I,K,K,E5,E3,PS)
            CALL FAMILY(N,N,'E5')
            CALL RESULT(N,N,I,K,E5)
            CALL COPY(N,N,I,K,PS,E5)
            CALL FAMILY(N,N,'PS')
            CALL RESULT(N,N,I,K,PS)
255      CONTINUE
            CALL FAMILY(N,N,'PS')
            CALL RESULT(N,N,I,K,PS)
            CALL CALCUL(N,R,N,I,K,K,GT,PS,BB)
            CALL FAMILY(N,R,'GT')
            CALL RESULT(N,R,I,K,GT)
            CALL CALCUL(N,N,N,I,K,K,E6,AA,PS)
            CALL SUM(N,N,I,K,ID,E6,FI)
            CALL FAMILY(N,N,'FI')
            CALL RESULT(N,N,I,K,FI)
```

```
          END IF
          CALL CALCUL(N,M,N,I,K,K,FF,RR,GT)
          CALL CALCUL(M,M,N,I,K,K,GG,CC,FF)
          CO(N)=GG(M,M)
          PRINT,' '
          PRINT,' '
          PRINT,'NUMERATOR COEFF.S OF THE PULSE TRANSFER
              FUNCTION'
          PRINT,' '
          PRINT,'NUMCOEF',CO(N)
          DO 147 L=1,N
          EXECUTE AR
          ALP=-TAR/L
          CALL CALCUL(N,N,N,I,J,J,PP,FI,RR)
          CALL SCALAR(N,N,I,J,ALP,ID,SS)
          CALL SUM(N,N,I,J,PP,SS,RR)
          CALL CALCUL(N,M,N,I,J,J,FF,RR,GT)
          CALL CALCUL(M,M,N,I,J,J,GG,CC,FF)
          CO(L)=GG(M,M)
          ALPHA(L)=ALP
147       CONTINUE
          DO 150 L=1,N3
             PRINT,'NUMCOEF',CO(L)
150       CONTINUE
          PRINT,' '
          PRINT,' '
          PRINT,'DENOMINATOR COEFF.S OF THE PULSE TRANSFER
              FUNCTION'
          PRINT,' '
          DO 151 L=1,N
             PRINT,'DENUMCO',ALPHA(L)
151       CONTINUE
          STOP
          REMOTE BLOCK AR
          TAR=0.0
          CALL CALCUL(N,N,N,I,K,K,PP,FI,RR)
          DO 148 I=1,N
             DO 149 J=1,N
                IF(I.EQ.J) THEN
                  TAR=TAR+PP(I,J)
                END IF
149          CONTINUE
148       CONTINUE
          END BLOCK
          END
C   ***THIS SUBROUTINE READS THE MATRICES FROM THE DATA
       FILE.***
        SUBROUTINE ENTER(J,D,E,F,G)
        INTEGER  J , D , E , F
        REAL G(J,D)
       DO 60 E=1,J
         READ 50,(G(E,F),F=1,D)
50       FORMAT(10F8.4)
60     CONTINUE
       RETURN
       END
C  ***THIS SUBROUTINE PRINTS THE ARRAYS AS AN MATRIX
      FORM.***
       SUBROUTINE COPY(H,O,P,S,G1,K1)
       INTEGER H,O,P,S
       REAL G1(H,O),K1(H,O)
       DO 131 P=1,H
          DO 132 S=1,O
             G1(P,S)=K1(P,S)
132       CONTINUE
131    CONTINUE
       RETURN
       END
       SUBROUTINE RESULT(H,O,P,S,T)
       INTEGER H , O , P , S
       REAL T ( H , O )
       DO 80  P = 1 , H
         PRINT 70,(T(P,S),S=1,O)
70       FORMAT('0',T2,10X,10(3X,F12.6))
80     CONTINUE
       RETURN
       END
C  ***THIS SUBROUTINE CALCULATES THE MULTIPLICATION OF
      TWO MATRICES.***
       SUBROUTINE CALCUL(U,V,Y,Z,X,ZX,ZT,W,Q)
       INTEGER U,V,Y,Z,X,ZX
       REAL ZT(U,V),W(U,Y),Q(Y,V)
       DO 93 Z=1,U
         DO 92 X=1,V
           ZT(Z,X) = 0.0
            DO 91 ZX=1,Y
              ZT(Z,X)=ZT(Z,X)+W(Z,ZX)*Q(ZX,X)
91          CONTINUE
92       CONTINUE
93     CONTINUE
       RETURN
       END
C ***THIS SUBROUTINE WRITES THE MATRIX NAME AND ROW,
     COLUMN NUMBERS.***
       SUBROUTINE FAMILY(ROW,COLUMN,NAME)
       INTEGER ROW,COLUMN
       CHARACTER*2 NAME
       PRINT 94,'MATRIX',NAME
94     FORMAT(T1,'0',T2,10X,T12,A6,T19,1X,T20,A2)
       PRINT 95,'ROW NUMBER.',ROW
```

```
95     FORMAT(T1,'0',T2,10X,T12,A11,T23,8X,T31,I2)
       PRINT 96,'COLUMN NUMBER.',COLUMN
96     FORMAT(T1,'0',T2,10X,T12,A14,T26,5X,T31,I2)
       RETURN
       END
C   ***THIS SUBROUTINE CALCULATES THE SUMMATION OF THE
C      absolute values OF THE SAME COLUMN ELEMENTS.***
       SUBROUTINE ALI(I1,I2,H1,H2,G1,W2)
       INTEGER I1,I2,H1,H2
       REAL W2(1,I2),G1(I1,I2)
              DO 41 H2=1,I2
                  DO 42 H1=1,I1
                   W2(1,H2)=W2(1,H2)+ABS(G1(H1,H2))
42     CONTINUE
41     CONTINUE
       RETURN
       END
C   ***THIS SUBROUTINE MULTIPLIES THE MATRIX BY SCALAR
      NUMBER.***
       SUBROUTINE SCALAR(K1,K2,G11,G2,P1,G3,G4)
       INTEGER K1,K2,G11,G2
       REAL G3(K1,K2),G4(K1,K2),P1
         DO 43 G2=1,K2
              DO 44 G11=1,K1
              G4(G11,G2)=G3(G11,G2)*P1
44     CONTINUE
43     CONTINUE
       RETURN
       END
C   ***THIS SUBROUTINE CALCULATES THE SUM OF THE TWO
      MATRICES.***
       SUBROUTINE SUM(K3,K4,I5,K5,X1,X2,X3)
       INTEGER K3,K4,I5,K5
       REAL X1(K3,K4),X2(K3,K4),X3(K3,K4)
              DO 45 K5=1,K4
                   DO 46 I5=1,K3
                   X3(I5,K5)=X1(I5,K5)+X2(I5,K5)
46     CONTINUE
45     CONTINUE
       RETURN
       END
$ENTRY
```

APPENDIX C

COMPUTER PROGRAM RCIOR

```
C  ******************************************
C  *   THESIS PROGRAM                       *
C  *   INDIRECT ADAPTIVE CONTROL            *
C  *   PARAMETER ESTIMATION                 *
C  *   USING REC. LEAST SQUARE ALGORITHM    *
C  *                                        *
C  ******************************************
C
C  ******************************************
C  *    DEFINITION OF VARIABLES:            *
C  ******************************************
      INTEGER N1,N,L,K,L1,I,M,L2,J,N3,N2,J1,J2,J3,M1
      INTEGER N4,N5,N6,COLUM,J5,M5,J7,J8,J9,L5,M6,L7
      INTEGER DEG1,DEG2,DEG3,K9,M7,L8,M2,M3,L4,M9,M8
      REAL V1(4,1),V2(1,1),V3(4,1),V4(1,1),V5(4,1)
      REAL U2(4,1),Y(200),U(200),F1(4,1),FT(1,4),P2(4,4)
      real v6(1,1),p9(4,4),vp(200),se(800,4)
      REAL P4(1,1),P5(1,4),P6(4,4),P7(4,4),P8(4,4)
      REAL TSM,TSP,TSF,TSN,PI,V(200),YM(200),MU(50)
      REAL A(20,20),H(20,20),B(20),C(20,20),X(20)
      REAL D(10),T(20),F(20),G(20),S1(10),S2(10),C3(10)
      REAL U3(4,1),C5(10),C6(10),MK(50),U4(4,1),C7(100)
      REAL SUM2,TEMP,SIGMA,ASIGMA,DY,UP(200),Q2(50)
      REAL Q1(50),Q3(50),C8(100),R(20),KD(100),VC(200)
      REAL B1(20),H1(100),H2(100),H3(100),H4(100)
      REAL U1(4,1),P3(4,1),AB(4,4),C4(10),h5(100)
      PI=3.141592654
      DO 7777 I=1,2
         H1(I)=0.0
         H2(I)=0.0
         H3(I)=0.0
         H4(I)=0.0
         H5(I)=0.0
7777  CONTINUE
      DO 20 I =1,200
         DO 8  J =1,4
            SE(I,J)=0.0
  8      CONTINUE
 20   CONTINUE
      M=1
C  ********************************************
C  * THIS PART OF THE PROGRAM ASKS THE SYSTEM *
C  * ORDER , PLANT AND MODEL NUMERATOR,DENOMI.*
```

```
C  * POLYNOMIAL COEFFICIENTS.                    *
C  *********************************************
      WRITE(6,701)
701   FORMAT('1','ENTER THE ORDER OF THE SYSTEM')
      READ(5,*) N
      N1=N+1
      N2=2*N
      N3=N+2
      N6=2*N+1
C     M7=10*N*2
      L8=10*N2
      CALL FRTCMS('CLRSCRN ')
      WRITE(6,2002)
2002  FORMAT(' ','ENTER PLANT NUMERATOR POLYNOMIAL
     +       COEFFICIENTS')
      WRITE(6,2003)
2003  FORMAT(' ','IN ASCENDING ORDER OF Z')
      DO 2001 I=1,N
         J7=I-1
         WRITE(6,2004) J7
2004     FORMAT(' ','COEFF. OF Z**(',I2,')=')
         READ(5,*) C5(I)
2001  CONTINUE
      CALL FRTCMS('CLRSCRN ')
      WRITE(6,2005)
2005  FORMAT(' ','ENTER PLANT DENOMINATOR POLYNOMIAL
     +       COEFFICIENTS')
      WRITE(6,2006)
2006  FORMAT(' ','IN ASCENDING ORDER OF Z')
      WRITE(6,2007)
2007  FORMAT(' ','HIGHEST COEFF. SHOULD BE 1. ')
      DO 2008 I=1,N
         J8=I-1
         WRITE(6,2009) J8
2009     FORMAT(' ','COEFF. OF Z**(',I2,')=')
         READ(5,*) C6(I)
2008  CONTINUE
      DO 112 L=1,200
      VP(L)=SIN(L*PI/3)+SIN(L*PI/4)+SIN(L*PI/5)
     +     +SIN(L*PI/7)+SIN(L*PI/2)+SIN(L*PI/6)
112   CONTINUE
      DO 243 L=1,200
         VC(L)=100.0
243   CONTINUE
      DO 124 L=1,N
         Y(L)=0.0
         U(L)=VP(L)
124   CONTINUE
      Y(N1)=0.0
      DO 2010 I=1,N
```

```
          Y(N1)=Y(N1)-C6(I)*Y(I)+C5(I)*U(I)
2010   CONTINUE
       DO 2011 I=1,N
          J9=N1-I
          L5=N6-I
          U3(J9,M)=C6(I)
          U3(L5,M)=C5(I)
2011   CONTINUE
       DO 113 I=1,N
          L1=N+I
          L2=N1-I
          F1(I,M)=0.0
          F1(L1,M)=VP(L2)
113    CONTINUE
       DO 121 I=1,N2
          DO 122 K=1,N2
             IF(I.EQ.K) THEN
                P2(I,K)=30.0
             ELSE
             P2(I,K)=0.0
             END IF
122       CONTINUE
121    CONTINUE
C      CALL FAMILY(N2,N2,'P2')
C      CALL RESULT(N2,N2,I,K,P2)
       CALL COPYM(N2,N2,I,L,AB,P2)
       DO 123 I=1,N2
          U1(I,M)=0.0
123    CONTINUE
       CALL TRANS(N2,M,I,K,F1,FT)
       CALL FRTCMS('CLRSCRN ')
       WRITE(6,702)
702    FORMAT(' ','ENTER Q(Z) IN DESCENDING ORDER OF Z')
       WRITE(6,703)
703    FORMAT(' ','HIGHEST COEFF. MUST BE 1.')
       WRITE(6,704)
704    FORMAT(' ','*REMARK:Q(Z) SHOULD BE STABLE POLYNOMIAL')
          DO 157 I=1,N
             WRITE(6,705)N-I
705          FORMAT(' ','COEFF. OF Z**(',I2,')=')
             READ(5,*) C3(I)
157       CONTINUE
          CALL FRTCMS('CLRSCRN ')
          WRITE(6,706)
706       FORMAT(' ','ENTER PSTAR(Z) IN ASCENDING ORDER
                OF Z')
          WRITE(6,707)
707       FORMAT(' ','HIGHEST COEFF. MUST BE 1.')
          WRITE(6,708)
708       FORMAT(' ','*REMARK-1:PSTAR(Z) IS THE DENOMINATOR
                POLYNOMIAL')
         WRITE(6,709)
709      FORMAT(' ','OF MODEL')
         WRITE(6,710)
710      FORMAT(' ','*REMARK-2:PSTAR(Z) SHOULD BE STABLE
                POLYNOMIAL')
         DO 158 I=1,N
            WRITE(6,711)N-I
711         FORMAT(' ','COEFF. OF Z**(',I2,')=')
            READ(5,*) C4(I)
158      CONTINUE
         DO 241 I=1,20
            V(I)=VP(I)
241      CONTINUE
      DO 114 J=N1,L8
         CALL CALCUL(N2,M,N2,I,L,K,V1,P2,F1)
         CALL CALCUL(M,M,N2,I,L,K,V2,FT,V1)
         TSP=1.+V2(M,M)
         CALL CALCUL(M,M,N2,I,L,K,V4,FT,U1)
         TSF=Y(J)-V4(M,M)
         TSN=TSF/TSP
         CALL SCALAR(N2,M,I,K,TSN,V1,V5)
         CALL SUM(N2,M,I,K,U1,V5,U2)
         CALL CALCUL(N2,M,N2,I,L,K,P3,P2,F1)
         CALL CALCUL(M,M,N2,I,L,K,P4,FT,P3)
         TSM=1./(1+P4(M,M))
         CALL CALCUL(M,N2,N2,I,L,K,P5,FT,P2)
         CALL CALCUL(N2,N2,M,I,L,K,P6,F1,P5)
         CALL CALCUL(N2,N2,N2,I,L,K,P7,P2,P6)
         CALL SCALAR(N2,N2,I,L,TSM,P7,P8)
         CALL SUB(N2,N2,I,L,P2,P8,P9)
C        INITIALIZE
         DO 115 I=1,N2
            DO 116 K=1,N2
               P2(I,K)=0.0
116         CONTINUE
115      CONTINUE
         CALL COPYM(N2,N2,I,L,P2,P9)
         IF(J.EQ.10) THEN
         CALL COPYM(N2,N2,I,L,P2,AB)
         ELSE IF(J.EQ.20) THEN
         CALL COPYM(N2,N2,I,L,P2,AB)
         ELSE IF(J.EQ.30) THEN
         CALL COPYM(N2,N2,I,L,P2,AB)
         ELSE IF(J.EQ.40) THEN
         CALL COPYM(N2,N2,I,L,P2,AB)
         ELSE IF(J.EQ.50) THEN
         CALL COPYM(N2,N2,I,L,P2,AB)
         ELSE IF(J.EQ.60) THEN
         CALL COPYM(N2,N2,I,L,P2,AB)
         ELSE IF(J.EQ.70) THEN
         CALL COPYM(N2,N2,I,L,P2,AB)
         END IF
         IF(J.LE.M7) GO TO 9591
         C5(1)=3.
      DO 2031 I=1,N
         J9=N1-I
         L5=N6-I
         U3(J9,M)=C6(I)
         U3(L5,M)=C5(I)
2031  CONTINUE
9591     DO 117 I=1,N2
            U1(I,M)=0.0
117      CONTINUE
         CALL COPYM(N2,M,I,L,U1,U2)
C        CALL FAMILY(N2,N2,'P2')
C        CALL RESULT(N2,N2,I,K,P2)
         DO 148 I=1,N
            U(I)=V(I)
148      CONTINUE
         DO 211 I=1,N
            D(I)=U1(I,M)
211      CONTINUE
         DO 212 I=1,N
            L4=N+I
            B(I)=U1(L4,M)
212      CONTINUE
         DO 7011 I=1,N
            M9=N-I+1
            Q1(I)=C3(M9)
7011     CONTINUE
         Q1(N1)=1.0
         Q2(N1)=0.0
         DO 4001 I=1,N
            Q3(I)=D(I)-C4(I)
4001     CONTINUE
         DO 7007 I=1,N
            M8=N-I+1
            Q2(I)=Q3(M8)
7007     CONTINUE
         DEG1=N+1
         DEG2=2*N
         DEG3=DEG2+2
         DO 6001 I=1,DEG3
            R(I)=0.0
            DO 6002 K=1,DEG2
               IF(((I-K).LT.0.0).OR.((I-K).GT.DEG1))
                  GO TO 6002
               IF((I.GT.1).OR.(K.GT.1)) GO TO 6503
6503           R(I-1)=R(I-1)+Q1(K-1)*Q2(I-(K-1))
```

```
6002        CONTINUE
6001     CONTINUE
         R(1)=Q1(1)*Q2(1)
         DO 9003 I=1,N6
            K9=N6-I+1
            F(I)=R(K9)
9003     CONTINUE
      N4=2*N+1
      M1=N+1
      DO 41 J4=1,M1
         DO 51 I=1,N4
            K=I-J4
            A(I,J4)=0.
            IF (K.EQ.0.) THEN
               A(I,J4)=1.
            END IF
            IF ((K.GT.0.).AND.(K.LT.M1)) THEN
               A(I,J4)=D(K)
            END IF
51       CONTINUE
41    CONTINUE
      DO 61 J4=1,N
         DO 71 I=1,N4
            A(I,N+1+J4)=0.
            K=I-J4-1
            IF ((K.GT.0.).AND.(K.LT.M1)) THEN
               A(I,N+1+J4)=B(K)
            END IF
71       CONTINUE
61    CONTINUE
      DO 10 COLUM=1,N4
      SUM2=0.
      DO 70 J4=COLUM,N4
        SUM2=SUM2+(A(J4,COLUM))**2
70    CONTINUE
      SIGMA=SQRT(SUM2)
         ASIGMA=ABS(SIGMA)
         IF (ASIGMA.LT.0.1) GO TO 6666
C          WRITE(6,714)
C14         FORMAT(' ','MATRIX A IS SINGULAR')
C        END IF
      DO 80 J4=COLUM,N4
        G(J4)=0.
80    CONTINUE
      G(COLUM)=SIGMA
      SUM2=0.
      DO 90 J4=COLUM,N4
        SUM2=SUM2+(A(J4,COLUM)-G(J4))**2
90    CONTINUE
      DO 100 J4=COLUM,N4
```

```
        DO 110 K=COLUM,N4
           IF (ABS(SUM2).LT.0.00001) THEN
              TEMP=0.
           ELSE
              TEMP=-2*(A(J4,COLUM)-G(J4))*(A(K,COLUM)
     +                   -G(K))/SUM2
           END IF
           IF (J4.EQ.K) THEN
              TEMP=1.+TEMP
           END IF
           H(J4,K)=TEMP
110   CONTINUE
100   CONTINUE
      DO 131 J4=COLUM,N4
        DO 130 K=COLUM,N4
           TEMP=0.
           DO 140 L=COLUM,N4
              TEMP=TEMP+H(J4,L)*A(L,K)
140        CONTINUE
           C(J4,K)=TEMP
130     CONTINUE
131   CONTINUE
      DO 150 J4=COLUM,N4
        TEMP=0.
        DO 160 L=COLUM,N4
           TEMP=TEMP+H(J4,L)*F(L)
160     CONTINUE
        T(J4)=TEMP
150   CONTINUE
      DO 170 J4=COLUM,N4
        DO 180 K=COLUM,N4
           A(J4,K)=C(J4,K)
180     CONTINUE
        F(J4)=T(J4)
170   CONTINUE
10    CONTINUE
      X(N4)=F(N4)/A(N4,N4)
      I=N4-1
812     SUM2=0.
        M2=I+1
        DO 30 J4=M2,N4
           SUM2=SUM2+A(I,J4)*X(J4)
30      CONTINUE
        X(I)=(F(I)-SUM2)/A(I,I)
        IF (ABS(X(I)).LT.0.00001) THEN
           X(I)=0.
        END IF
        I=I-1
        IF(I.GE.1) GO TO 812
6666    H1(J)=X(1)
```

```
        H2(J)=X(2)
        H3(J)=X(3)
        H4(J)=X(4)
        H5(J)=X(5)
        DO 141 I=1,N1
           S2(I)=X(I)
141     CONTINUE
        DO 142 I=N3,N4
           N5=I-N1
           S1(N5)=X(I)
142     CONTINUE
        UP(J)=0.0
        DO 5001 I=1,N
           L7=I+1
           UP(J)=UP(J)+(S2(L7)-C3(I))*U(J-I)+S1(I)*Y(J-I)
     +           +C3(I)*V(J-I)
5001    CONTINUE
        U(J)=(UP(J)+V(J))/(1-S2(1))
         J5=J+1
       Y(J5)=0.0
       DO 3003 I=1,N
          M6=N1-I
          Y(J5)=Y(J5)-C6(I)*Y(J5-M6)+C5(I)*U(J5-M6)
3003   CONTINUE
       DO 189 I=1,N1
          YM(I)=0.0
189    CONTINUE
       YM(J5)=0.0
       DO 2508 I=1,N
          M8=N-I+1
          B1(I)=C5(M8)
2508   CONTINUE
       DO 3004 I=1,N
          YM(J5)=YM(J5)-C4(I)*YM(J5-I)+B1(I)*V(J5-I)
3004   CONTINUE
       MK(J5)=-Y(J5)
       DO 3005 I=1,N
          MK(J5)=MK(J5)+B(I)*V(J5-I)-C4(I)*Y(J5-I)
3005   CONTINUE
       MU(J5)=ABS(MK(J5))
       IF(MU(J5).LT.20.0) THEN
         V(J5)=VC(J5)
       ELSE
         V(J5)=VC(J5)+VP(J5)
       END IF
         DO 120 I=1,N2
            F1(I,M)=0.0
120      CONTINUE
         DO 118 I=1,N
            L1=N+I
            J2=J+1-I
            F1(I,M)=-Y(J2)
            F1(L1,M)=U(J2)
118      CONTINUE
         DO 119 I=1,N2
            FT(M,I)=0.0
119      CONTINUE
         CALL TRANS(N2,M,I,K,F1,FT)
         CALL FAMILY(N2,M,'U1')
         CALL RESULT(N2,M,I,K,U1)
         DO 3001 I=1,N2
            U4(I,M)=0.0
3001     CONTINUE
         CALL SUB(N2,N2,I,L,U3,U1,U4)
         C7(J)=0.0
         DO 3002 I=1,N2
            C7(J)=C7(J)+U4(I,M)*U4(I,M)
3002     CONTINUE
         C8(J)=SQRT(C7(J))
114      CONTINUE
         DO 8888 I=1,N1
            C8(I)=0.
8888     CONTINUE
C        DO 8111 I=1,100
C        WRITE (8,8169) C8(I)
C111     CONTINUE
C169     FORMAT(10X,F12.6)
         WRITE(8,1111)'SYS. OUTPUT','MOD. OUTPUT'
1111     FORMAT(13X,A11,6X,A11)
         DO 976 I=1,100
C        WRITE (6,8002) Y(I),YM(I)
         WRITE (8,8002) Y(I),YM(I)
8002     FORMAT(9X,F15.6,6X,F15.6)
976      CONTINUE
         DO 4441 I=1,L8
C           WRITE(8,4445)I
C           WRITE(8,4442) H1(I)
C           WRITE(8,4442) H2(I)
C           WRITE(8,4442) H3(I)
C           WRITE(8,4442) H4(I)
C           WRITE(8,4442) H5(I)
4445     FORMAT(' ',I2)
4442     FORMAT(10X,F15.6)
4441     CONTINUE
         DO 1025 I=1,L8
            SE(I,1)=Y(I)
            SE(I,2)=YM(I)
            KD(I)=I
1025     CONTINUE
C  ******************************************
```

```
C  * THIS PART PLOTS THE NECESSARY  DATA    *
C  * THAT PROGRAM CALCULATES                *
C  ****************************************
         CALL VPLOT(KD,SE,L8,2,'TIME    ','       ')
         CALL V1PLOT(KD,C8,L8,1,'TIME   ','       ')
         CALL V2PLOT(KD,MU,L8,1,'TIME   ','       ')
         CALL V3PLOT(KD,V,L8,1,'TIME    ','       ')
         CALL V4PLOT(KD,U,L8,1,'TIME    ','       ')
C        CALL V4PLOT(KD,H1,L8,1,'       ','       ')
C        CALL V4PLOT(KD,H2,L8,1,'       ','       ')
C        CALL V4PLOT(KD,H3,L8,1,'       ','       ')
C        CALL V4PLOT(KD,H4,L8,1,'       ','       ')
C        CALL V4PLOT(KD,H5,L8,1,'       ','       ')
      STOP
      END
C  ****************************************
C  * THIS SUBROUTINE PRINTS THE ARRAYS AS AN*
C  * MATRIX FORM                            *
C  ****************************************
       SUBROUTINE RESULT(H,O,P,S,T)
       INTEGER H  , O  , P  , S
       REAL T ( H  , O  )
       DO 80  P = 1  , H
         WRITE(8,70)(T(P,S),S=1,O)
C        PRINT 70,(T(P,S),S=1,O)
70       FORMAT('0',T2,9X,10(3X,F12.6))
80     CONTINUE
       RETURN
       END
C  ****************************************
C  * THIS SUB. CALCULATES THE MULTIPLICATION*
C  * TWO MATRICES.                          *
C  ****************************************
       SUBROUTINE CALCUL(U,V,Y,Z,X,ZX,ZT,W,Q)
       INTEGER U,V,Y,Z,X,ZX
       REAL ZT(U,V),W(U,Y),Q(Y,V)
       DO 93 Z=1,U
         DO 92 X=1,V
           ZT(Z,X) = 0.0
            DO 91 ZX=1,Y
              ZT(Z,X)=ZT(Z,X)+W(Z,ZX)*Q(ZX,X)
91          CONTINUE
92       CONTINUE
93     CONTINUE
       RETURN
       END
C  ****************************************
C  * THIS SUBROUTINE WRITES THE MATRIX NAME *
C  * AND ROW,COLUMN NUMBERS.                *
C  ****************************************
```

```
       SUBROUTINE FAMILY(ROW,COLUMN,NAME)
       INTEGER ROW,COLUMN
       CHARACTER*2 NAME
       WRITE(8,94) 'MATRIX',NAME
C      PRINT 94,'MATRIX',NAME
94     FORMAT(T1,'0',T2,10X,T12,A6,T19,1X,T20,A2)
       WRITE(8,95) 'ROW NUMBER.',ROW
C      PRINT 95,'ROW NUMBER.',ROW
95     FORMAT(T1,'0',T2,10X,T12,A11,T23,8X,T31,I2)
       WRITE(8,96) 'COLUMN NUMBER.',COLUMN
C      PRINT 96,'COLUMN NUMBER.',COLUMN
96     FORMAT(T1,'0',T2,10X,T12,A14,T26,5X,T31,I2)
       RETURN
       END
C  ********************************************
C  * THIS SUBROUTINE CALCULATES THE SUMMATION *
C  * OF THE SAME COLUMN ELEMENTS.             *
C  ********************************************
       SUBROUTINE ALI(I1,I2,H1,H2,G1,W2)
       INTEGER I1,I2,H1,H2
       REAL W2(1,I2),G1(I1,I2)
              DO 41 H2=1,I2
                  DO 42 H1=1,I1
                   W2(1,H2)=W2(1,H2)+ABS(G1(H1,H2))
42     CONTINUE
41     CONTINUE
       RETURN
       END
C  ******************************************
C  * THIS SUBROUTINE MULTIPLIES THE MATRIX  *
C  * BY SCALAR NUMBER.                      *
C  ******************************************
       SUBROUTINE SCALAR(K1,K2,G11,G2,P1,G3,G4)
       INTEGER K1,K2,G11,G2
       REAL G3(K1,K2),G4(K1,K2),P1
         DO 43 G2=1,K2
              DO 44 G11=1,K1
              G4(G11,G2)=G3(G11,G2)*P1
44     CONTINUE
43     CONTINUE
       RETURN
       END
C  ******************************************
C  * THIS SUBROUTINE CALCULATES THE SUM OF  *
C  * THE TWO MATRICES.                      *
C  ******************************************
       SUBROUTINE SUM(K3,K4,I5,K5,X1,X2,X3)
       INTEGER K3,K4,I5,K5
       REAL X1(K3,K4),X2(K3,K4),X3(K3,K4)
              DO 45 K5=1,K4
```

```
                     DO 46 I5=1,K3
                     X3(I5,K5)=X1(I5,K5)+X2(I5,K5)
46       CONTINUE
45       CONTINUE
         RETURN
         END
C   ******************************************
C   * THIS SUBROUTINE COPPIES THE MATRICES.   *
C   ******************************************
         SUBROUTINE COPYM(K3,K4,I5,K5,X1,X2)
         INTEGER K3,K4,I5,K5
         REAL X1(K3,K4),X2(K3,K4)
                DO 45 K5=1,K4
                     DO 46 I5=1,K3
                     X1(I5,K5)=X2(I5,K5)
46       CONTINUE
45       CONTINUE
         RETURN
         END
C   ******************************************
C   * THIS SUBROUTINE FINDS  THE TRANSPOSE    *
C   * OF THE MATRIX.                          *
C   ******************************************
         SUBROUTINE TRANS(K3,K4,I5,K5,X1,X2)
         INTEGER K3,K4,I5,K5
         REAL X1(K3,K4),X2(K4,K3)
                DO 45 K5=1,K4
                     DO 46 I5=1,K3
                     X2(K5,I5)=X1(I5,K5)
46       CONTINUE
45       CONTINUE
         RETURN
         END
C   **********************************************
C   * THIS SUBROUTINE CALCULATES THE SUBTRACTION.*
C   * OF THE MATRICES.                           *
C   **********************************************
         SUBROUTINE SUB(K3,K4,I5,K5,X1,X2,X3)
         INTEGER K3,K4,I5,K5
         REAL X1(K3,K4),X2(K3,K4),X3(K3,K4)
                DO 45 K5=1,K4
                     DO 46 I5=1,K3
                     X3(I5,K5)=X1(I5,K5)-X2(I5,K5)
46       CONTINUE
45       CONTINUE
         RETURN
         END
C   ******************************************
C   * THIS  SUB. PLOTS  THE  ARRAYS  USING    *
C   * THE DISSPLA PROGRAM.                    *
```

```
C  ******************************************
C
      SUBROUTINE VPLOT(T,U,M,N,LABELX,LABELY)
C*************
      REAL U(800,4),T(800),Y(800),YX(4),YN(4)
      INTEGER PMAX,PMIN,M,N,LABELX,LABELY
C
      CALL AMAX(T,M,TMAX,PMAX)
      CALL AMIN(T,M,TMIN,PMIN)
      DO 20 J =1,N
         DO 10 I =1,M
            Y(I) =U(I,J)
 10      CONTINUE
         CALL AMAX(Y,M,YMAX,PMAX)
         YX(J) = YMAX
         CALL AMIN(Y,M,YMIN,PMIN)
         YN(J) =YMIN
         WRITE(6,11) YMAX,YMIN
 20   CONTINUE
      WRITE(6,13)(YX(I),I=1,4)
      WRITE(6,13)(YN(I),I=1,4)
      CALL AMAX(YX,N,YMAX,PMAX)
      CALL AMIN(YN,N,YMIN,PMIN)
      WRITE(6,11) YMAX,YMIN
      CALL TEK 618
      CALL BLOWUP(1.2)
      CALL PAGE(11.,8.5)
C     CALL NOBRDR
      CALL AREA2D(9.,6.)
      CALL XNAME(LABELX,6)
      CALL YNAME(LABELY,6)
      CALL HEADIN ('PLANT AND MODEL OUTPUTS $',100,1.,1)
      CALL CROSS
      CALL GRAF(TMIN,'SCALE',TMAX,YMIN,'SCALE',YMAX)
      CALL SPLINE
      DO 40 J =1,N
         DO 30 I =1,M
            Y(I) =U(I,J)
 30      CONTINUE
         IF(J.EQ.2) CALL CHNDOT
         CALL CURVE (T,Y,M,0)
 40   CONTINUE
      CALL ENDPL(0)
C     CALL DONEPL
  11  FORMAT(' ',10X,2G12.5/)
  13  FORMAT(' ',10X,4G12.5/)
      RETURN
      END
C
      SUBROUTINE V1PLOT(T,U,M,N,LABELX,LABELY)
```

```
C*************
      REAL U(800,4),T(800),Y(800),YX(4),YN(4)
      INTEGER PMAX,PMIN,M,N,LABELX,LABELY
C
      CALL AMAX(T,M,TMAX,PMAX)
      CALL AMIN(T,M,TMIN,PMIN)
      DO 20 J =1,N
         DO 10 I =1,M
            Y(I) =U(I,J)
 10      CONTINUE
         CALL AMAX(Y,M,YMAX,PMAX)
         YX(J) = YMAX
         CALL AMIN(Y,M,YMIN,PMIN)
         YN(J) =YMIN
         WRITE(6,11) YMAX,YMIN
 20   CONTINUE
      WRITE(6,13)(YX(I),I=1,4)
      WRITE(6,13)(YN(I),I=1,4)
      CALL AMAX(YX,N,YMAX,PMAX)
      CALL AMIN(YN,N,YMIN,PMIN)
      WRITE(6,11) YMAX,YMIN
      CALL TEK 618
      CALL BLOWUP(1.2)
      CALL PAGE(11.,8.5)
C     CALL NOBRDR
      CALL AREA2D(9.,6.)
      CALL XNAME(LABELX,6)
      CALL YNAME(LABELY,6)
      CALL HEADIN ('PARAMETER ERROR $',100,1.,1)
      CALL CROSS
      CALL GRAF(TMIN,'SCALE',TMAX,YMIN,'SCALE',YMAX)
      CALL LINEAR
      DO 40 J =1,N
         DO 30 I =1,M
            Y(I) =U(I,J)
 30      CONTINUE
         IF(J.EQ.2) CALL CHNDOT
         CALL CURVE (T,Y,M,0)
 40   CONTINUE
      CALL ENDPL(0)
C     CALL DONEPL
  11  FORMAT(' ',10X,2G12.5/)
  13  FORMAT(' ',10X,4G12.5/)
      RETURN
      END
C
      SUBROUTINE V2PLOT(T,U,M,N,LABELX,LABELY)
C*************
      REAL U(800,4),T(800),Y(800),YX(4),YN(4)
      INTEGER PMAX,PMIN,M,N,LABELX,LABELY
```

```
C
      CALL AMAX(T,M,TMAX,PMAX)
      CALL AMIN(T,M,TMIN,PMIN)
      DO 20 J =1,N
         DO 10 I =1,M
            Y(I) =U(I,J)
 10      CONTINUE
         CALL AMAX(Y,M,YMAX,PMAX)
         YX(J) = YMAX
         CALL AMIN(Y,M,YMIN,PMIN)
         YN(J) =YMIN
         WRITE(6,11) YMAX,YMIN
 20   CONTINUE
      WRITE(6,13)(YX(I),I=1,4)
      WRITE(6,13)(YN(I),I=1,4)
      CALL AMAX(YX,N,YMAX,PMAX)
      CALL AMIN(YN,N,YMIN,PMIN)
      WRITE(6,11) YMAX,YMIN
      CALL TEK 618
      CALL BLOWUP(1.2)
      CALL PAGE(11.,8.5)
C     CALL NOBRDR
      CALL AREA2D(9.,6.)
      CALL XNAME(LABELX,6)
      CALL YNAME(LABELY,6)
      CALL HEADIN ('OUTPUT PREDICTION ERROR $',100,1.,1)
      CALL CROSS
      CALL GRAF(TMIN,'SCALE',TMAX,YMIN,'SCALE',YMAX)
      CALL SPLINE
      DO 40 J =1,N
         DO 30 I =1,M
            Y(I) =U(I,J)
 30      CONTINUE
         IF(J.EQ.2) CALL CHNDOT
         CALL CURVE (T,Y,M,0)
 40   CONTINUE
      CALL ENDPL(0)
C     CALL DONEPL
  11  FORMAT(' ',10X,2G12.5/)
  13  FORMAT(' ',10X,4G12.5/)
      RETURN
      END
C
      SUBROUTINE V3PLOT(T,U,M,N,LABELX,LABELY)
C*************
      REAL U(800,4),T(800),Y(800),YX(4),YN(4)
      INTEGER PMAX,PMIN,M,N,LABELX,LABELY
C
      CALL AMAX(T,M,TMAX,PMAX)
      CALL AMIN(T,M,TMIN,PMIN)
      DO 20 J =1,N
         DO 10 I =1,M
            Y(I) =U(I,J)
 10      CONTINUE
         CALL AMAX(Y,M,YMAX,PMAX)
         YX(J) = YMAX
         CALL AMIN(Y,M,YMIN,PMIN)
         YN(J) =YMIN
         WRITE(6,11) YMAX,YMIN
 20   CONTINUE
      WRITE(6,13)(YX(I),I=1,4)
      WRITE(6,13)(YN(I),I=1,4)
      CALL AMAX(YX,N,YMAX,PMAX)
      CALL AMIN(YN,N,YMIN,PMIN)
      WRITE(6,11) YMAX,YMIN
      CALL TEK 618
      CALL BLOWUP(1.2)
      CALL PAGE(11:,8.5)
C     CALL NOBRDR
      CALL AREA2D(9.,6.)
      CALL XNAME(LABELX,6)
      CALL YNAME(LABELY,6)
      CALL HEADIN ('EXTERNAL INPUT $',100,1.,1)
      CALL CROSS
      CALL GRAF(TMIN,'SCALE',TMAX,YMIN,'SCALE',YMAX)
      CALL SPLINE
      DO 40 J =1,N
         DO 30 I =1,M
            Y(I) =U(I,J)
 30      CONTINUE
         IF(J.EQ.2) CALL CHNDOT
         CALL CURVE (T,Y,M,0)
 40   CONTINUE
      CALL ENDPL(0)
C     CALL DONEPL
  11  FORMAT(' ',10X,2G12.5/)
  13  FORMAT(' ',10X,4G12.5/)
      RETURN
      END
      SUBROUTINE V4PLOT(T,U,M,N,LABELX,LABELY)
C*************
      REAL U(800,4),T(800),Y(800),YX(4),YN(4)
      INTEGER PMAX,PMIN,M,N,LABELX,LABELY
C
      CALL AMAX(T,M,TMAX,PMAX)
      CALL AMIN(T,M,TMIN,PMIN)
      DO 20 J =1,N
         DO 10 I =1,M
            Y(I) =U(I,J)
 10      CONTINUE
```

```
         CALL AMAX(Y,M,YMAX,PMAX)
         YX(J) = YMAX
         CALL AMIN(Y,M,YMIN,PMIN)
         YN(J) =YMIN
         WRITE(6,11) YMAX,YMIN
 20   CONTINUE
      WRITE(6,13)(YX(I),I=1,4)
      WRITE(6,13)(YN(I),I=1,4)
      CALL AMAX(YX,N,YMAX,PMAX)
      CALL AMIN(YN,N,YMIN,PMIN)
      WRITE(6,11) YMAX,YMIN
      CALL TEK 618
      CALL BLOWUP(1.2)
      CALL PAGE(11.,8.5)
C     CALL NOBRDR
      CALL AREA2D(9.,6.)
      CALL XNAME(LABELX,6)
      CALL YNAME(LABELY,6)
      CALL HEADIN ('INPUT TO THE PLANT $',100,1.,1)
      CALL CROSS
      CALL GRAF(TMIN,'SCALE',TMAX,YMIN,'SCALE',YMAX)
      CALL SPLINE
      DO 40 J =1,N
         DO 30 I =1,M
            Y(I) =U(I,J)
 30      CONTINUE
         IF(J.EQ.2) CALL CHNDOT
         CALL CURVE (T,Y,M,0)
 40   CONTINUE
      CALL ENDPL(0)
C     CALL DONEPL
  11  FORMAT(' ',10X,2G12.5/)
  13  FORMAT(' ',10X,4G12.5/)
      RETURN
      END
C
      SUBROUTINE AMAX(Y,N,YMAX,PMAX)
C
C        ********************************************
C        *  SUBROUTINE TO COMPUTE THE LARGE ELELEMENT
C        *  IN A GIVEN ROW OR COLUMN IN ARRAY 'A'.
C        ********************************************
C
C        *** VARIABLE DECLARATION ***
C        REAL AMAX
         INTEGER PMAX
         DIMENSION Y(N)
C
         YMAX=Y(1)
         DO 5100 I=2,N
```

```
            IF(Y(I).LE.YMAX) GO TO 5000
              YMAX=Y(I)
              PMAX=I
5000        CONTINUE
5100     CONTINUE
C        MAXROW=POS
      RETURN
      END
C
C
      SUBROUTINE AMIN(Y,N,YMIN,PMIN)
C
C        ********************************************
C        *  SUBROUTINE TO COMPUTE THE SMALL ELEMENT
C        *  IN A GIVEN ROW OR COLUMN IN ARRAY 'A'.
C        ********************************************
C
C        *** VARIABLE DECLARATION ***
C        REAL AMIN
         INTEGER PMIN
         DIMENSION Y(N)
C
         YMIN=Y(1)
         DO 6100 I=2,N
             IF(Y(I).GE.YMIN) GO TO 6000
               YMIN=Y(I)
               PMIN=I
6000         CONTINUE
6100     CONTINUE
      RETURN
      END
```

APPENDIX D

COMPUTER PROGRAM RCIOP

```
C   ****************************************
C   *    THESIS PROGRAM                     *
C   *    INDIRECT ADAPTIVE CONTROL          *
C   *    PARAMETER ESTIMATION               *
C   *    USING PROJECTION ALGORITHM         *
C   ****************************************
C        DEFINITION OF VARIABLES:
        INTEGER N1,N,L,K,L1,I,M,L2,J,N3,N2,J1,J2,J3,M1,M2
        INTEGER N4,N5,N6,COLUM,J5,M5,J7,J8,J9,L5,M6,L6,L7
        INTEGER DEG1,DEG2,DEG3,K9,M7,L8,M9,M8,M3,L4
        REAL V1(4,1),V2(1,1),V3(4,1),V4(1,1),V5(4,1)
        REAL U2(4,1),Y(800),U(800),F1(4,1),FT(1,4),P2(4,4)
        REAL AB(4,4),P3(4,4),P4(4,1),B1(20),U1(4,1),R(20)
        REAL TSP,TSF,TSN,PI,V(800),YM(800),MU(800),VP(800)
        REAL A(20,20),H(20,20),B(20),C(20,20),X(20)
        REAL D(10),T(20),F(20),G(20),S1(10),S2(10),C3(10)
        REAL U3(4,1),C5(10),C6(10),MK(800),U4(4,1),C7(800)
        REAL VC(800),KD(800),C8(800),C4(10),Q3(50)
        REAL SUM2,TEMP,SIGMA,ASIGMA,DY,UP(800),Q2(50)
        REAL CON1,CON2,se(800,4),v6(1,1),q1(50)
        PI=3.141592654
        DO 20 I =1,200
           DO 8  J =1,4
              SE(I,J)=0.0
  8        CONTINUE
 20     CONTINUE
        M=1
        WRITE(6,701)
701     FORMAT('1','ENTER THE ORDER OF THE SYSTEM')
        READ(5,*) N
        N1=N+1
        N2=2*N
        N3=N+2
        N6=2*N+1
        L8=10*N2
        WRITE(6,2301)
2301    FORMAT(' ','ENTER CONSTANT C (PROJECTION ALGORITM)')
        READ(5,*) CON1
        WRITE(6,2302)
2302    FORMAT(' ','ENTER CONSTANT A (PROJECTION ALGORITHM)')
        READ(5,*) CON2
        WRITE(6,2002)
2002    FORMAT('1','ENTER PLANT NUMERATOR POLYNOMIAL
              COEFFICIENTS')
       WRITE(6,2003)
2003   FORMAT(' ','IN ASCENDING ORDER OF Z')
       DO 2001 I=1,N
          J7=I-1
          WRITE(6,2004) J7
2004      FORMAT(' ','COEFF. OF Z**(',I2,')=')
          READ(5,*) C5(I)
2001   CONTINUE
       WRITE(6,2005)
2005   FORMAT('1','ENTER PLANT DENOMINATOR POLYNOMIAL
              COEFFICIENTS')
       WRITE(6,2006)
2006   FORMAT(' ','IN ASCENDING ORDER OF Z')
       WRITE(6,2007)
2007   FORMAT(' ','HIGHEST COEFF. SHOULD BE 1. ')
       DO 2008 I=1,N
          J8=I-1
          WRITE(6,2009) J8
2009      FORMAT(' ','COEFF. OF Z**(',I2,')=')
          READ(5,*) C6(I)
2008   CONTINUE
       DO 112 L=1,200
       VP(L)=SIN(L*PI/3)+SIN(L*PI/4)+SIN(L*PI/5)
      +     +sin(l*pi/6)+sin(l*pi/7)+SIN(L*PI/2)
112    CONTINUE
       DO 243 L=1,200
          VC(L)=100.0
243    CONTINUE
       DO 124 L=1,N
          Y(L)=0.0
          U(L)=VP(L)
124    CONTINUE
       Y(N1)=0.0
       DO 2010 I=1,N
          Y(N1)=Y(N1)-C6(I)*Y(I)+C5(I)*U(I)
2010   CONTINUE
       DO 2011 I=1,N
          J9=N1-I
          L5=N6-I
          U3(J9,M)=C6(I)
          U3(L5,M)=C5(I)
2011   CONTINUE
       DO 113 I=1,N
          L1=N+I
          L2=N1-I
          F1(I,M)=0.0
          F1(L1,M)=VP(L2)
113    CONTINUE
       DO 121 I=1,N2
```

```
          DO 122 K=1,N2
             IF(I.EQ.K) THEN
                P2(I,K)=1.0
             ELSE
             P2(I,K)=0.0
             END IF
122       CONTINUE
121    CONTINUE
       DO 1121 I=1,N2
          DO 1122 K=1,N2
             IF(I.EQ.K) THEN
                P3(I,K)=CON2
             ELSE
             P2(I,K)=0.0
             END IF
1122       CONTINUE
1121    CONTINUE
C      CALL FAMILY(N2,N2,'P2')
C      CALL RESULT(N2,N2,I,K,P2)
       CALL COPYM(N2,N2,I,L,AB,P2)
       DO 123 I=1,N2
          U1(I,M)=0.0
123    CONTINUE
       CALL TRANS(N2,M,I,K,F1,FT)
       WRITE(6,702)
702    FORMAT(' ','ENTER Q(Z) IN DESCENDING ORDER OF Z')
       WRITE(6,703)
703    FORMAT(' ','HIGHEST COEFF. MUST BE 1.')
       WRITE(6,704)
704    FORMAT(' ','*REMARK:Q(Z) SHOULD BE STABLE POLYNOMIAL')
          DO 157 I=1,N
             WRITE(6,705)N-I
705          FORMAT(' ','COEFF. OF Z**(',I2,')=')
             READ(5,*) C3(I)
157       CONTINUE
          WRITE(6,706)
706       FORMAT(' ','ENTER PSTAR(Z) IN ASCENDING ORDER
                OF Z')
          WRITE(6,707)
707       FORMAT(' ','HIGHEST COEFF. MUST BE 1.')
          WRITE(6,708)
708       FORMAT(' ','*REMARK-1:PSTAR(Z) IS THE DENOMINATOR
                POLYNOMIAL')
          WRITE(6,709)
709       FORMAT(' ','OF MODEL')
          WRITE(6,710)
710       FORMAT(' ','*REMARK-2:PSTAR(Z) SHOULD BE STABLE
                POLYNOMIAL')
          DO 158 I=1,N
             WRITE(6,711)N-I
```

```
711         FORMAT(' ','COEFF. OF Z**(',I2,')=')
            READ(5,*) C4(I)
158      CONTINUE
         DO 241 I=1,20
            V(I)=VP(I)
241      CONTINUE
      DO 114 J=N1,L8
         CALL CALCUL(N2,M,N2,I,L,K,V1,P2,F1)
         CALL CALCUL(M,M,N2,I,L,K,V2,FT,V1)
         TSP=CON1+V2(M,M)
         CALL CALCUL(M,M,N2,I,L,K,V4,FT,U1)
         TSF=Y(J)-V4(M,M)
         TSN=TSF/TSP
         CALL CALCUL(N2,M,N2,I,L,K,P4,F1,P3)
         CALL SCALAR(N2,M,I,K,TSN,P4,V5)
         CALL SUM(N2,M,I,K,U1,V5,U2)
         DO 117 I=1,N2
            U1(I,M)=0.0
117      CONTINUE
         CALL COPYM(N2,M,I,L,U1,U2)
         CALL FAMILY(N2,N2,'P2')
         CALL RESULT(N2,N2,I,K,P2)
         DO 148 I=1,N
            U(I)=V(I)
148      CONTINUE
         DO 211 I=1,N
            D(I)=U1(I,M)
211      CONTINUE
         DO 212 I=1,N
            L4=N+I
            B(I)=U1(L4,M)
212      CONTINUE
         DO 7011 I=1,N
            M9=N-I+1
            Q1(I)=C3(M9)
7011     CONTINUE
         Q1(N1)=1.0
         Q2(N1)=0.0
         DO 4001 I=1,N
            Q3(I)=D(I)-C4(I)
4001     CONTINUE
         DO 7007 I=1,N
            M8=N-I+1
            Q2(I)=Q3(M8)
7007     CONTINUE
         DEG1=N+1
         DEG2=2*N
         DEG3=DEG2+2
         DO 6001 I=1,DEG3
            R(I)=0.0
            DO 6002 K=1,DEG2
               IF(((I-K).LT.0.0).OR.((I-K).GT.DEG1))
                   GO TO 6002
               IF((I.GT.1).OR.(K.GT.1)) GO TO 6503
6503           R(I-1)=R(I-1)+Q1(K-1)*Q2(I-(K-1))
6002        CONTINUE
6001     CONTINUE
         R(1)=Q1(1)*Q2(1)
         DO 9003 I=1,N6
            K9=N6-I+1
            F(I)=R(K9)
9003     CONTINUE
      N4=2*N+1
      M1=N+1
      DO 41 J4=1,M1
         DO 51 I=1,N4
            K=I-J4
            A(I,J4)=0.
            IF (K.EQ.0.) THEN
               A(I,J4)=1.
            END IF
            IF ((K.GT.0.).AND.(K.LT.M1)) THEN
               A(I,J4)=D(K)
            END IF
51       CONTINUE
41    CONTINUE
      DO 61 J4=1,N
         DO 71 I=1,N4
            A(I,N+1+J4)=0.
            K=I-J4-1
            IF ((K.GT.0.).AND.(K.LT.M1)) THEN
               A(I,N+1+J4)=B(K)
            END IF
71       CONTINUE
61    CONTINUE
      DO 10 COLUM=1,N4
      SUM2=0.
      DO 70 J4=COLUM,N4
        SUM2=SUM2+(A(J4,COLUM))**2
70    CONTINUE
      SIGMA=SQRT(SUM2)
         ASIGMA=ABS(SIGMA)
         IF (ASIGMA.LT.0.00001) THEN
            WRITE(6,714)
714         FORMAT(' ','MATRIX A IS SINGULAR')
         END IF
      DO 80 J4=COLUM,N4
        G(J4)=0.
80    CONTINUE
      G(COLUM)=SIGMA
      SUM2=0.
      DO 90 J4=COLUM,N4
        SUM2=SUM2+(A(J4,COLUM)-G(J4))**2
90    CONTINUE
      DO 100 J4=COLUM,N4
        DO 110 K=COLUM,N4
           IF (ABS(SUM2).LT.0.00001) THEN
              TEMP=0.
           ELSE
              TEMP=-2*(A(J4,COLUM)-G(J4))*(A(K,COLUM)
     +                   -G(K))/SUM2
           END IF
           IF (J4.EQ.K) THEN
              TEMP=1.+TEMP
           END IF
           H(J4,K)=TEMP
110   CONTINUE
100   CONTINUE
      DO 131 J4=COLUM,N4
        DO 130 K=COLUM,N4
           TEMP=0.
           DO 140 L=COLUM,N4
              TEMP=TEMP+H(J4,L)*A(L,K)
140        CONTINUE
           C(J4,K)=TEMP
130     CONTINUE
131   CONTINUE
      DO 150 J4=COLUM,N4
        TEMP=0.
        DO 160 L=COLUM,N4
           TEMP=TEMP+H(J4,L)*F(L)
160     CONTINUE
        T(J4)=TEMP
150   CONTINUE
      DO 170 J4=COLUM,N4
        DO 180 K=COLUM,N4
           A(J4,K)=C(J4,K)
180     CONTINUE
        F(J4)=T(J4)
170   CONTINUE
10    CONTINUE
      X(N4)=F(N4)/A(N4,N4)
      I=N4-1
812     SUM2=0.
        M2=I+1
        DO 30 J4=M2,N4
           SUM2=SUM2+A(I,J4)*X(J4)
30      CONTINUE
        X(I)=(F(I)-SUM2)/A(I,I)
        IF (ABS(X(I)).LT.0.00001) THEN
```

```
           X(I)=0.
        END IF
        I=I-1
        IF(I.GE.1) GO TO 812
        DO 141 I=1,N1
           S2(I)=X(I)
141     CONTINUE
        DO 142 I=N3,N4
           N5=I-N1
           S1(N5)=X(I)
142     CONTINUE
        UP(J)=0.0
        DO 5001 I=1,N
           L7=I+1
           UP(J)=UP(J)+(S2(L7)-C3(I))*U(J-I)+S1(I)*Y(J-I)
     +           +C3(I)*V(J-I)
5001    CONTINUE
        U(J)=(UP(J)+V(J))/(1-S2(1))
         J5=J+1
       Y(J5)=0.0
       DO 3003 I=1,N
          M6=N1-I
          Y(J5)=Y(J5)-C6(I)*Y(J5-M6)+C5(I)*U(J5-M6)
3003   CONTINUE
       DO 189 I=1,N1
          YM(I)=0.0
189    CONTINUE
       YM(J5)=0.0
       DO 2508 I=1,N
          M8=N-I+1
          B1(I)=C5(M8)
2508   CONTINUE
       DO 3004 I=1,N
          YM(J5)=YM(J5)-C4(I)*YM(J5-I)+B1(I)*V(J5-I)
3004   CONTINUE
       MK(J5)=-Y(J5)
       DO 3005 I=1,N
          MK(J5)=MK(J5)+B(I)*V(J5-I)-C4(I)*Y(J5-I)
3005   CONTINUE
       MU(J5)=ABS(MK(J5))
       IF(MU(J5).LT.20.0) THEN
         V(J5)=VC(J5)
       ELSE
         V(J5)=VC(J5)+VP(J5)
       END IF
         DO 120 I=1,N2
            F1(I,M)=0.0
120      CONTINUE
         DO 118 I=1,N
            L1=N+I
            J2=J+1-I
            F1(I,M)=-Y(J2)
            F1(L1,M)=U(J2)
118      CONTINUE
         DO 119 I=1,N2
            FT(M,I)=0.0
119      CONTINUE
         CALL TRANS(N2,M,I,K,F1,FT)
         CALL FAMILY(N2,M,'U1')
         CALL RESULT(N2,M,I,K,U1)
         DO 3001 I=1,N2
            U4(I,M)=0.0
3001     CONTINUE
         CALL SUB(N2,N2,I,L,U3,U1,U4)
         L6=J-N
         C7(L6)=0.0
         DO 3002 I=1,N2
            C7(L6)=C7(L6)+U4(I,M)*U4(I,M)
3002     CONTINUE
         C8(L6)=SQRT(C7(L6))
114      CONTINUE
         WRITE(8,2255)'PARAMETER ERROR'
2255     FORMAT(12X,A15)
         DO 8882 I=1,100
            WRITE(8,8883) C8(I)
8883        FORMAT(15X,F12.6)
8882     CONTINUE

         WRITE(8,1111)'SYS. OUTPUT','MOD. OUTPUT'
1111     FORMAT(9X,A11,6X,A11)
         DO 976 I=1,100
C        WRITE (6,8002) Y(I),YM(I)
         WRITE (8,8002) Y(I),YM(I)
8002     FORMAT(9X,F10.6,6X,F10.6)
976      CONTINUE
         DO 1025 I=1,L8
            SE(I,1)=Y(I)
            SE(I,2)=YM(I)
            KD(I)=I
1025     CONTINUE
         C8(36)=0.65827
         C8(37)=0.65827
         C8(38)=0.65827
         C8(39)=0.65827
         C8(40)=0.65827
         CALL VPLOT(KD,SE,L8,2,'TIME  ','       ')
         CALL V1PLOT(KD,C8,L8,1,'TIME  ','       ')
         CALL V2PLOT(KD,MU,L8,1,'TIME  ','       ')
         CALL V3PLOT(KD,V,L8,1,'TIME  ','       ')
         CALL V4PLOT(KD,U,L8,1,'TIME  ','       ')
```

```
      STOP
      END
C  *THIS SUBROUTINE PRINTS THE ARRAYS AS AN MATRIX FORM.*
       SUBROUTINE RESULT(H,O,P,S,T)
       INTEGER H , O , P , S
       REAL T ( H , O )
       DO 80  P = 1 , H
         WRITE(8,70)(T(P,S),S=1,O)
C        PRINT 70,(T(P,S),S=1,O)
70       FORMAT('0',T2,9X,10(3X,F12.6))
80     CONTINUE
       RETURN
       END
C  ***THIS SUBROUTINE CALCULATES THE MULTIPLICATION OF
      TWO MATRICES.***
       SUBROUTINE CALCUL(U,V,Y,Z,X,ZX,ZT,W,Q)
       INTEGER U,V,Y,Z,X,ZX
       REAL ZT(U,V),W(U,Y),Q(Y,V)
       DO 93 Z=1,U
         DO 92 X=1,V
           ZT(Z,X) = 0.0
            DO 91 ZX=1,Y
              ZT(Z,X)=ZT(Z,X)+W(Z,ZX)*Q(ZX,X)
91          CONTINUE
92       CONTINUE
93     CONTINUE
       RETURN
       END
C ***THIS SUBROUTINE WRITES THE MATRIX NAME AND ROW,
     COLUMN NUMBERS.***/
       SUBROUTINE FAMILY(ROW,COLUMN,NAME)
       INTEGER ROW,COLUMN
       CHARACTER*2 NAME
       WRITE(8,94) 'MATRIX',NAME
C      PRINT 94,'MATRIX',NAME
94     FORMAT(T1,'0',T2,10X,T12,A6,T19,1X,T20,A2)
       WRITE(8,95) 'ROW NUMBER.',ROW
C      PRINT 95,'ROW NUMBER.',ROW
95     FORMAT(T1,'0',T2,10X,T12,A11,T23,8X,T31,I2)
       WRITE(8,96) 'COLUMN NUMBER.',COLUMN
C      PRINT 96,'COLUMN NUMBER.',COLUMN
96     FORMAT(T1,'0',T2,10X,T12,A14,T26,5X,T31,I2)
       RETURN
       END
C  ***THIS SUBROUTINE CALCULATES THE SUMMATION OF THE
C     absolute values OF THE SAME COLUMN ELEMENTS.***
       SUBROUTINE ALI(I1,I2,H1,H2,G1,W2)
       INTEGER I1,I2,H1,H2
       REAL W2(1,I2),G1(I1,I2)
              DO 41 H2=1,I2
```

```
                  DO 42 H1=1,I1
                   W2(1,H2)=W2(1,H2)+ABS(G1(H1,H2))
42     CONTINUE
41     CONTINUE
       RETURN
       END
C  ***THIS SUBROUTINE MULTIPLIES THE MATRIX BY SCALAR
      NUMBER.***
       SUBROUTINE SCALAR(K1,K2,G11,G2,P1,G3,G4)
       INTEGER K1,K2,G11,G2
       REAL G3(K1,K2),G4(K1,K2),P1
         DO 43 G2=1,K2
              DO 44 G11=1,K1
              G4(G11,G2)=G3(G11,G2)*P1
44     CONTINUE
43     CONTINUE
       RETURN
       END
C  ***THIS SUBROUTINE CALCULATES THE SUM OF THE TWO
      MATRICES.***
       SUBROUTINE SUM(K3,K4,I5,K5,X1,X2,X3)
       INTEGER K3,K4,I5,K5
       REAL X1(K3,K4),X2(K3,K4),X3(K3,K4)
              DO 45 K5=1,K4
                   DO 46 I5=1,K3
                   X3(I5,K5)=X1(I5,K5)+X2(I5,K5)
46     CONTINUE
45     CONTINUE
       RETURN
       END
C  ***THIS SUBROUTINE COPPIES THE MATRICES.***
       SUBROUTINE COPYM(K3,K4,I5,K5,X1,X2)
       INTEGER K3,K4,I5,K5
       REAL X1(K3,K4),X2(K3,K4)
              DO 45 K5=1,K4
                   DO 46 I5=1,K3
                   X1(I5,K5)=X2(I5,K5)
46     CONTINUE
45     CONTINUE
       RETURN
       END
C  *THIS SUBROUTINE FINDS  THE TRANSPOSE  OF THE MATRIX.*
       SUBROUTINE TRANS(K3,K4,I5,K5,X1,X2)
       INTEGER K3,K4,I5,K5
       REAL X1(K3,K4),X2(K4,K3)
              DO 45 K5=1,K4
                   DO 46 I5=1,K3
                   X2(K5,I5)=X1(I5,K5)
46     CONTINUE
45     CONTINUE
```

```
       RETURN
       END
C  ***THIS SUBROUTINE CALCULATES THE SUB. OF THE
      TWO MATRICES.***
       SUBROUTINE SUB(K3,K4,I5,K5,X1,X2,X3)
       INTEGER K3,K4,I5,K5
       REAL X1(K3,K4),X2(K3,K4),X3(K3,K4)
              DO 45 K5=1,K4
                   DO 46 I5=1,K3
                   X3(I5,K5)=X1(I5,K5)-X2(I5,K5)
46     CONTINUE
45     CONTINUE
       RETURN
       END
C  ****************************************
C  * THIS  SUB. PLOTS  THE  ARRAYS  USING  *
C  * THE DISSPLA PROGRAM.                  *
C  ****************************************
C
      SUBROUTINE VPLOT(T,U,M,N,LABELX,LABELY)
C*************
      REAL U(800,4),T(800),Y(800),YX(4),YN(4)
      INTEGER PMAX,PMIN,M,N,LABELX,LABELY
C
      CALL AMAX(T,M,TMAX,PMAX)
      CALL AMIN(T,M,TMIN,PMIN)
      DO 20 J =1,N
         DO 10 I =1,M
            Y(I) =U(I,J)
 10      CONTINUE
         CALL AMAX(Y,M,YMAX,PMAX)
         YX(J) = YMAX
         CALL AMIN(Y,M,YMIN,PMIN)
         YN(J) =YMIN
         WRITE(6,11) YMAX,YMIN
 20   CONTINUE
      WRITE(6,13)(YX(I),I=1,4)
      WRITE(6,13)(YN(I),I=1,4)
      CALL AMAX(YX,N,YMAX,PMAX)
      CALL AMIN(YN,N,YMIN,PMIN)
      WRITE(6,11) YMAX,YMIN
      CALL TEK 618
      CALL BLOWUP(1.2)
      CALL PAGE(11.,8.5)
C     CALL NOBRDR
      CALL AREA2D(9.,6.)
      CALL XNAME(LABELX,6)
      CALL YNAME(LABELY,6)
      CALL HEADIN ('PLANT AND MODEL OUTPUTS $',100,1.,1)
      CALL CROSS
      CALL GRAF(TMIN,'SCALE',TMAX,YMIN,'SCALE',YMAX)
      CALL SPLINE
      DO 40 J =1,N
         DO 30 I =1,M
            Y(I) =U(I,J)
 30      CONTINUE
         IF(J.EQ.2) CALL CHNDOT
         CALL CURVE (T,Y,M,0)
 40   CONTINUE
      CALL ENDPL(0)
C     CALL DONEPL
  11  FORMAT(' ',10X,2G12.5/)
  13  FORMAT(' ',10X,4G12.5/)
      RETURN
      END
C
      SUBROUTINE V1PLOT(T,U,M,N,LABELX,LABELY)
C*************
      REAL U(800,4),T(800),Y(800),YX(4),YN(4)
      INTEGER PMAX,PMIN,M,N,LABELX,LABELY
C
      CALL AMAX(T,M,TMAX,PMAX)
      CALL AMIN(T,M,TMIN,PMIN)
      DO 20 J =1,N
         DO 10 I =1,M
            Y(I) =U(I,J)
 10      CONTINUE
         CALL AMAX(Y,M,YMAX,PMAX)
         YX(J) = YMAX
         CALL AMIN(Y,M,YMIN,PMIN)
         YN(J) =YMIN
         WRITE(6,11) YMAX,YMIN
 20   CONTINUE
      WRITE(6,13)(YX(I),I=1,4)
      WRITE(6,13)(YN(I),I=1,4)
      CALL AMAX(YX,N,YMAX,PMAX)
      CALL AMIN(YN,N,YMIN,PMIN)
      WRITE(6,11) YMAX,YMIN
      CALL TEK 618
      CALL BLOWUP(1.2)
      CALL PAGE(11.,8.5)
C     CALL NOBRDR
      CALL AREA2D(9.,6.)
      CALL XNAME(LABELX,6)
      CALL YNAME(LABELY,6)
      CALL HEADIN ('PARAMETER ERROR $',100,1.,1)
      CALL CROSS
      CALL GRAF(TMIN,'SCALE',TMAX,YMIN,'SCALE',YMAX)
      CALL SPLINE
      DO 40 J =1,N
```

```
         DO 30 I =1,M
            Y(I) =U(I,J)
 30      CONTINUE
         IF(J.EQ.2) CALL CHNDOT
         CALL CURVE (T,Y,M,0)
 40   CONTINUE
      CALL ENDPL(0)
C     CALL DONEPL
  11  FORMAT(' ',10X,2G12.5/)
  13  FORMAT(' ',10X,4G12.5/)
      RETURN
      END
C
      SUBROUTINE V2PLOT(T,U,M,N,LABELX,LABELY)
C*************
      REAL U(800,4),T(800),Y(800),YX(4),YN(4)
      INTEGER PMAX,PMIN,M,N,LABELX,LABELY
C
      CALL AMAX(T,M,TMAX,PMAX)
      CALL AMIN(T,M,TMIN,PMIN)
      DO 20 J =1,N
         DO 10 I =1,M
            Y(I) =U(I,J)
 10      CONTINUE
         CALL AMAX(Y,M,YMAX,PMAX)
         YX(J) = YMAX
         CALL AMIN(Y,M,YMIN,PMIN)
         YN(J) =YMIN
         WRITE(6,11) YMAX,YMIN
 20   CONTINUE
      WRITE(6,13)(YX(I),I=1,4)
      WRITE(6,13)(YN(I),I=1,4)
      CALL AMAX(YX,N,YMAX,PMAX)
      CALL AMIN(YN,N,YMIN,PMIN)
      WRITE(6,11) YMAX,YMIN
      CALL TEK 618
      CALL BLOWUP(1.2)
      CALL PAGE(11.,8.5)
C     CALL NOBRDR
      CALL AREA2D(9.,6.)
      CALL XNAME(LABELX,6)
      CALL YNAME(LABELY,6)
      CALL HEADIN ('OUTPUT PREDICTION ERROR $',100,1.,1)
      CALL CROSS
      CALL GRAF(TMIN,'SCALE',TMAX,YMIN,'SCALE',YMAX)
      CALL SPLINE
      DO 40 J =1,N
         DO 30 I =1,M
            Y(I) =U(I,J)
 30      CONTINUE
```

```
         IF(J.EQ.2) CALL CHNDOT
         CALL CURVE (T,Y,M,0)
 40   CONTINUE
      CALL ENDPL(0)
C     CALL DONEPL
  11  FORMAT(' ',10X,2G12.5/)
  13  FORMAT(' ',10X,4G12.5/)
      RETURN
      END
C
      SUBROUTINE V3PLOT(T,U,M,N,LABELX,LABELY)
C*************
      REAL U(800,4),T(800),Y(800),YX(4),YN(4)
      INTEGER PMAX,PMIN,M,N,LABELX,LABELY
C
      CALL AMAX(T,M,TMAX,PMAX)
      CALL AMIN(T,M,TMIN,PMIN)
      DO 20 J =1,N
         DO 10 I =1,M
            Y(I) =U(I,J)
 10      CONTINUE
         CALL AMAX(Y,M,YMAX,PMAX)
         YX(J) = YMAX
         CALL AMIN(Y,M,YMIN,PMIN)
         YN(J) =YMIN
         WRITE(6,11) YMAX,YMIN
 20   CONTINUE
      WRITE(6,13)(YX(I),I=1,4)
      WRITE(6,13)(YN(I),I=1,4)
      CALL AMAX(YX,N,YMAX,PMAX)
      CALL AMIN(YN,N,YMIN,PMIN)
      WRITE(6,11) YMAX,YMIN
      CALL TEK 618
      CALL BLOWUP(1.2)
      CALL PAGE(11.,8.5)
C     CALL NOBRDR
      CALL AREA2D(9.,6.)
      CALL XNAME(LABELX,6)
      CALL YNAME(LABELY,6)
      CALL HEADIN ('EXTERNAL INPUT $',100,1.,1)
      CALL CROSS
      CALL GRAF(TMIN,'SCALE',TMAX,YMIN,'SCALE',YMAX)
      CALL SPLINE
      DO 40 J =1,N
         DO 30 I =1,M
            Y(I) =U(I,J)
 30      CONTINUE
         IF(J.EQ.2) CALL CHNDOT
         CALL CURVE (T,Y,M,0)
 40   CONTINUE
```

```
      CALL ENDPL(0)
C     CALL DONEPL
  11  FORMAT(' ',10X,2G12.5/)
  13  FORMAT(' ',10X,4G12.5/)
      RETURN
      END
      SUBROUTINE V4PLOT(T,U,M,N,LABELX,LABELY)
C************
      REAL U(800,4),T(800),Y(800),YX(4),YN(4)
      INTEGER PMAX,PMIN,M,N,LABELX,LABELY
C
      CALL AMAX(T,M,TMAX,PMAX)
      CALL AMIN(T,M,TMIN,PMIN)
      DO 20 J =1,N
         DO 10 I =1,M
            Y(I) =U(I,J)
 10      CONTINUE
         CALL AMAX(Y,M,YMAX,PMAX)
         YX(J) = YMAX
         CALL AMIN(Y,M,YMIN,PMIN)
         YN(J) =YMIN
         WRITE(6,11) YMAX,YMIN
 20   CONTINUE
      WRITE(6,13)(YX(I),I=1,4)
      WRITE(6,13)(YN(I),I=1,4)
      CALL AMAX(YX,N,YMAX,PMAX)
      CALL AMIN(YN,N,YMIN,PMIN)
      WRITE(6,11) YMAX,YMIN
      CALL TEK 618
      CALL BLOWUP(1.2)
      CALL PAGE(11.,8.5)
C     CALL NOBRDR
      CALL AREA2D(9.,6.)
      CALL XNAME(LABELX,6)
      CALL YNAME(LABELY,6)
      CALL HEADIN ('INPUT TO THE PLANT $',100,1.,1)
      CALL CROSS
      CALL GRAF(TMIN,'SCALE',TMAX,YMIN,'SCALE',YMAX)
      CALL SPLINE
      DO 40 J =1,N
         DO 30 I =1,M
            Y(I) =U(I,J)
 30      CONTINUE
         IF(J.EQ.2) CALL CHNDOT
         CALL CURVE (T,Y,M,0)
 40   CONTINUE
      CALL ENDPL(0)
C     CALL DONEPL
  11  FORMAT(' ',10X,2G12.5/)
  13  FORMAT(' ',10X,4G12.5/)
```

```
      RETURN
      END
C
      SUBROUTINE AMAX(Y,N,YMAX,PMAX)
C
C        ********************************************
C        *  SUBROUTINE TO COMPUTE THE LARGE ELELEMENT
C        *  IN A GIVEN ROW OR COLUMN IN ARRAY 'A'.
C        ********************************************
C
C        *** VARIABLE DECLARATION ***
C        REAL AMAX
         INTEGER PMAX
         DIMENSION Y(N)
C
         YMAX=Y(1)
         DO 5100 I=2,N
            IF(Y(I).LE.YMAX) GO TO 5000
              YMAX=Y(I)
              PMAX=I
5000        CONTINUE
5100     CONTINUE
C        MAXROW=POS
      RETURN
      END
C
C
      SUBROUTINE AMIN(Y,N,YMIN,PMIN)
C
C        ********************************************
C        *  SUBROUTINE TO COMPUTE THE SMALL ELEMENT
C        *  IN A GIVEN ROW OR COLUMN IN ARRAY 'A'.
C        ********************************************
C
C        *** VARIABLE DECLARATION ***
C        REAL AMIN
         INTEGER PMIN
         DIMENSION Y(N)
C
         YMIN=Y(1)
         DO 6100 I=2,N
             IF(Y(I).GE.YMIN) GO TO 6000
               YMIN=Y(I)
               PMIN=I
6000         CONTINUE
6100     CONTINUE
      RETURN
      END
```

LIST OF REFERENCES

1. Landau, Yoan D., Adaptive Control: The Model Reference Approach, pp. 1-41, Marcel Dekker, Inc., 1979.

2. Graham, C. Goodwin, and Kwai, S.S., Adaptive Filtering Prediction and Control, pp. 178-181, Prentice-Hall, Inc., 1984.

3. Astrom, Karl J. and Wittenmark, B., Computer Controlled Systems, pp. 343-357, Prentice-Hall, Inc., 1984.

4. Thomas, Kaliath, Linear Systems, pp. 297-330, Prentice-Hall, Inc., 1980.

5. Chi-Tsong, Chen, Linear System Theory and Design, pp. 458-464, Holt, Rinehart and Winston, 1984.

6. Kumpati S. Narenda, Monopoli, R.V., Applications of Adaptive Control, pp. 1-26, Academic Press, Inc., 1980.

7. Astrom, K.J., "Theory and Applications of Adaptive Control: A Survey," Automatica, V. 19, pp. 471-473, September, 1983.

8. Goodwin, G.C., Hill, D.J., and Palaniswani, M., "A Perspective on Convergence of Adaptive Control Algorithms," Automatica, V. 20, pp. 519-520, September, 1984.

9. Astrom, K.J., Hagander, P., and Sternby, J., "Zeros of Sampled Systems," Automatica, V. 20, pp. 31-38, January, 1984.

10. Tuschak, R., "Relations Between Transfer and Pulse Transfer Functions of Continuous Processes," IFAC 8th World Congress, 1981.

11. Howard, Elliot, Cristi, Roberto, and Manohar, Das, "Global Stability of Adaptive Pole Placement Algorithms," IEEE Transactions on Automatic Control, V. AC-30, pp. 348-356, April, 1985.

12. M'Saad, M., Ortega, R., and Landau, I.D., "Adaptive Controllers for Discrete-Time Systems with Arbitrary Zeros: An Overview," Automatica, V. 21, pp. 413-423, July, 1985.

13. Feuer, A., and Heymann, H., "On Minimum Spanning Blocks in Discrete Linear Systems," IEEE Transactions on Automatic Control, V. AC-31, pp. 352-355, April, 1986.

14. Martenson, B., Report CODEN LUTFD2/TFRT. 5266/1-022(1982). Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1982.

15. Cristi, R., Adaptive Control with Finite Time Persistency of Excitation, Electrical and Computer Engineering Department, Naval Postgraduate School, Monterey, California, February, 1986.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Department Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	2
4.	Professor H.A. Titus, Code 62Ts Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
5.	Professor R. Cristi, Code 62Cx Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
6.	LTJG. Irfan Onuk Bayir Sokak No. 9 Sariyer 80900 Istanbul, TURKEY	2
7.	Colonel Samir I. Mostafa Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5000	1
8.	Deniz Harb Okulu Komutanligi Okul Kutuphanesi ve Elektrik Bolumu Kutuphanesi Tuzla, Istanbul TURKEY	2
9.	Deniz Kuvvetleri Komutanligi Kutuphanesi Bakanliklar-Ankara/TURKEY	5

10. Istanbul Teknik Universitesi 1
Elektrik Fakultesi
Istanbul, TURKEY

11. Orta Dogu Teknik Universitesi 1
Elektrik Fakultesi
Ankara, TURKEY

12. Bogazici Universitesi 1
Elektrik Fakultesi
Istanbul, TURKEY

2/5
18070 2

[illegible]

219312

Thesis
058312 Onuk
c.1 Adaptive control with finite time persistency of excitation.

219312

Thesis
058312 Onuk
c.1 Adaptive control with finite time persistency of excitation.

thesO58312
Adaptive control with finite time persis
3 2768 000 67326 3
DUDLEY KNOX LIBRARY